Control Software Engineering Approaches for Cyber-Physical Systems: A Systematic Mapping Study

AMAR BANERJEE and VENKATESH CHOPPELLA, International Institute of Information Technology, Hyderabad, India

Cyber-Physical Systems (CPS), robotics, the Internet of Things (IoT), and automotive systems are integral to modern technology. They are characterized by their safety criticality, accuracy, and real-time control requirements. Control software plays a crucial role in achieving these objectives by managing and coordinating the operations of various sub-systems. This article presents a novel Systematic Mapping Study (SMS) for control software engineering, analyzing 115 peer-reviewed papers. The study identifies, classifies, and maps existing solutions, providing a comprehensive and structured overview for practitioners and researchers. Our contributions include (1) a unique classification of literature into six research themes—engineering phases, engineering approaches, engineering paradigms, engineering artifacts, target application domains, and engineering concerns; (2) insights into the specificity of approaches to target technologies and phases; (3) the prominence of model-driven approaches for design and testing; (4) the lack of end-to-end engineering support in existing approaches; and (5) the emerging role of agile-based methods versus the dominance of waterfall-based methods. This article's significance lies in its thorough analysis and the high-level mapping of the solution space, offering new perspectives and a detailed roadmap for future research and innovation in control software engineering. The findings will guide advancements and best practices in the field, underscoring the article's impact.

CCS Concepts: • **General and reference** → **Surveys and overviews**; **Reference works**; *General literature*; • **Applied computing** → *Reference models*; • **Computer systems organization** → **Embedded and cyber-physical systems**; • **Information systems** → **Process control systems**; • **Computing methodologies** → *Control methods*;

Additional Key Words and Phrases: cyber-physical systems, mapping study, software engineering, control software, IoT, robotics

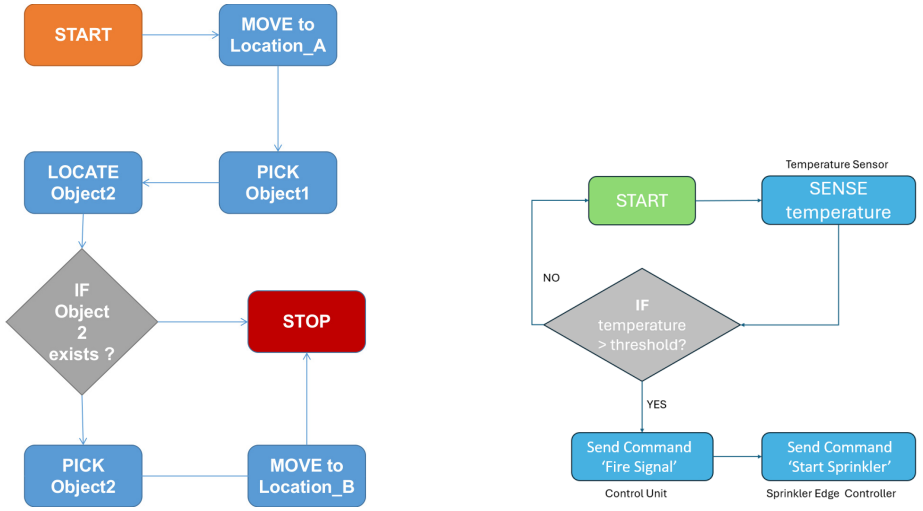
ACM Reference format:
Amar Banerjee and Venkatesh Choppella. 2025. Control Software Engineering Approaches for Cyber-Physical Systems: A Systematic Mapping Study. *ACM Trans. Cyber-Phys. Syst.* 9, 1, Article 6 (January 2025), 33 pages. https://doi.org/10.1145/3704737

1 Introduction

Cyber-Physical Systems (CPS) deliver operational goals through controllers orchestrating the underlying hardware and software components. Controllers may manifest as pure software (i.e.,

Authors' Contact Information: Amar Banerjee (corresponding author), International Institute of Information Technology, Hyderabad, India; e-mail: amar.banerjee@research.iiit.ac.in; Venkatesh Choppella, International Institute of Information Technology, Hyderabad, India; e-mail: venkatesh.choppella@iiit.ac.in.

ACM 2378-9638/2025/1-ART6
https://doi.org/10.1145/3704737



(a) A robot control flow for pick-n-place task (b) An IoT control flow for fire safety system

Fig. 1. Examples of control software.

sensor and actuator drivers), pure hardware (i.e., MCM controllers), or a hybrid form (e.g., PLCs). This article focuses on software controllers—referred to as *control software*. Control software is pivotal in CPS, robotics, the **Internet of Things (IoT)**, and automotive systems, enabling these technologies to execute complex operations effectively [8]. Figure 1 illustrates two instances of control flow as implemented by control software. Figure 1(a) depicts the tasks performed by a robot to move and pick up objects, whereas Figure 1(b) shows the asynchronous control flow within a fire safety system. The control software orchestrates these flows in different types of systems. Robust and accurate execution of the control software is imperative for a smooth and safe execution of the instruments.

The tragic Ethiopian air crash, resulting in the loss of all onboard, underscores the crucial importance of control software in maintaining safety and efficiency in complex systems. This incident, primarily linked to the **Maneuvering Characteristics Augmentation System (MCAS)** of the Boeing 737 MAX, highlights not just a flaw in a specific software component but exposes broader systemic issues in how control software integrates with overall system design and human factors [67]. The MCAS was designed to improve aircraft handling characteristics and decrease stall likelihood. Still, its aggressive intervention led to catastrophic outcomes based on erroneous sensor inputs and a lack of redundancy. Such incidents in the past have established the need for robust, correct and safe control software.

This study distinguishes itself by mapping existing control software methodologies and critically analyzing their applicability and effectiveness in different real-world scenarios. This approach aims to bridge the theoretical and practical aspects of control software engineering, offering insights into how these methodologies can be tailored for diverse industrial applications. While there is extensive research in control software engineering, a notable gap exists in its practical application, particularly in adapting these methodologies to varied industrial needs. This study addresses this gap by mapping the existing literature and providing a pragmatic perspective on how these methodologies can be effectively implemented in different industrial technologies.

1.1 Real-Life Concerns for Control Software

Control software in CPS plays a crucial role across diverse domains, each requiring unique characteristics to address specific challenges. Here, we provide detailed insights into control software applications in various specialized contexts:

Fire Safety. Control software ensures continuous monitoring and rapid response to fire hazards in fire safety applications. It integrates with various sensors and alarm systems, processing real-time data to detect anomalies and trigger automated responses such as activating sprinklers or alerting emergency services. The software's reliability and security are paramount to prevent failures that could endanger lives and property. It must operate under stringent regulations to ensure immediate and accurate hazard detection and response, minimizing false alarms and maximizing safety.

Warehouse Automation. In warehouse automation, control software manages robotic systems to handle tasks such as picking, packing, sorting, and inventory management. The software integrates with sensors and data sources to monitor inventory levels and operational conditions in real-time. It coordinates multiple autonomous robots, avoiding collisions and optimizing routes using advanced algorithms for path planning and task scheduling. High reliability and fault tolerance are essential to prevent operational disruptions, while security measures protect against unauthorized access and cyber-attacks. The software's adaptability to dynamic environments and its efficiency in managing fluctuating inventory levels and order volumes are critical for seamless warehouse operations [119].

Large Physical Instruments. Control software for large physical instruments like the **Square Kilometer Array (SKA)** manages the orchestration of thousands of individual antennae, ensuring precise synchronization and data collection. It handles vast amounts of data with high availability, robust real-time processing, and fault tolerance. This software must ensure continuous operation even in the event of hardware failures, making it critical for achieving the scientific goals of astronomical observations. The ability to process and analyze large datasets in real-time allows for immediate adjustments and calibrations, ensuring the accuracy and reliability of collected data [57].

Aviation. Aviation control software requires precise real-time actuation and high fault tolerance to manage multiple flight-related tasks. It ensures immediate responsiveness to flight dynamics, such as changes in speed and altitude, and handles emergencies like engine failures. Integrating with avionics systems, the software provides pilots with critical information and automated controls, enhancing flight safety and efficiency. The software must adhere to stringent safety standards and undergo rigorous testing to ensure reliability under all conditions [113].

Medical and Healthcare. Control software must achieve high precision and real-time adaptability in the medical field, especially in robotic and remote surgeries. It controls robotic surgical instruments to perform delicate procedures accurately, ensuring patient safety. The software must comply with strict regulatory standards for safety and reliability, provide secure data transmission, and maintain seamless operation under various conditions. Its ability to adapt to real-time feedback during surgeries is crucial for successful outcomes [121, 126].

Logistics and Transportation. In logistics, control software like that used in Amazon's delivery systems supports secure and efficient package delivery. It optimizes real-time delivery routes, adapts to traffic conditions, and ensures timely deliveries. The software manages warehouse inventory and coordinates with various transportation modes, streamlining the logistics chain. High reliability and security measures protect against disruptions and unauthorized access, ensuring the integrity of the delivery process [67].

While commonalities exist, such as the need for security and real-time operations, control software requirements and implementations vary significantly across different sectors. This variability reflects each sector's unique challenges, underscoring the importance of developing domain-specific, context-aware control software to meet diverse operational demands. As such, the development of control software must be context-aware, catering to the specialized needs of each domain to ensure efficacy and safety.

1.2 Related Work

Existing studies on CPS have significantly advanced our understanding of interconnected technologies, enhancing operational efficiency and ensuring interoperability across diverse industries. These studies have explored various aspects such as architecture [70], security [148], standards, algorithms [28], engineering paradigms [12], applications [92], and verification [37]. However, these contributions often do not specifically focus on control software, which acts as the nervous system of CPS [11].

Our research builds upon this foundational work by conducting a **Systematic Mapping Study (SMS)** that uniquely identifies and analyzes emerging trends and gaps in control software engineering within CPS. The novelty of this research lies in its comprehensive classification and mapping of the literature into distinct research themes, providing new insights and a structured overview that was previously lacking.

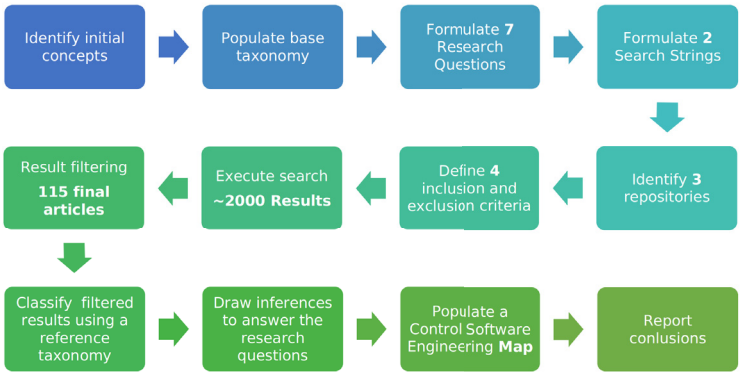
The selected domains for this study—CPS, IoT, robotics, and Automotive systems—represent the forefront of control software applications, crucial for enhancing safety, efficiency, and innovation. These areas, chosen for their critical significance [37, 89], face unique challenges central to advancing control software engineering. Our study addresses these challenges by focusing on strategic imperatives of Industry 4.0, which emphasizes the integration of advanced technologies [8, 26, 48] to foster intelligent operations.

By including a variety of application categories, from "large physical instruments" like the SKA to essential safety systems in "fire safety," our research highlights the versatility and necessity of sophisticated control software methodologies. This approach ensures the study covers a wide array of scenarios where the effectiveness and dependability of control software are critical, aligning with the study's goals to examine its broad applicability across different contexts and scales. The unique contribution of this study lies in its holistic view of control software engineering, providing a roadmap for future research and innovation in this dynamic field.

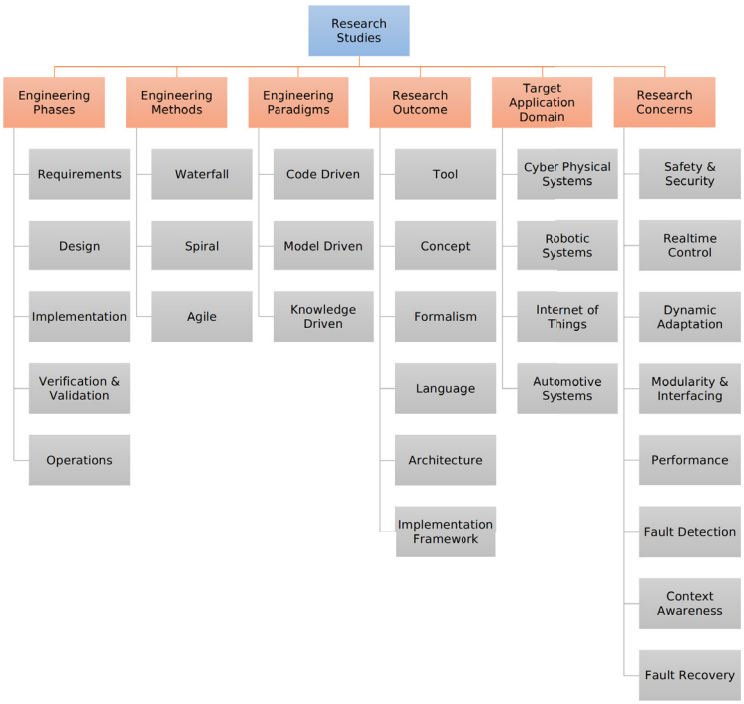
This mapping study is reported in five sections. Section 2 describes the procedure used to conduct this mapping study. Section 3 maps the literature approaches against the taxonomy categories and derives inferences from the data. The articles are then analyzed to answer the **Research Questions (RQs)** and identify where these approaches fit in the control software engineering lifecycle. Section 4 discusses the results and highlights the challenges and research opportunities. Finally, Section 5 concludes the mapping study, followed by a brief discussion on future work.

2 Survey Procedure

This SMS focuses on identifying the approaches along the dimensions like engineering phases, techniques, concerns, application domains, and so on. We modify the method proposed by Petersen et al. [103] by adding an initial step to populate a base taxonomy, which captures our initial understanding of the field of study. All possible RQs (based on the base taxonomy) are derived by drawing relations between the various elements in the base taxonomy. We choose a subset of RQs from all possible questions. Finally, the standard SMS practice is followed to identify and categorize articles, extract data, analyze, and report the result. Figure 2(a) illustrates a high-level view of the survey procedure.



(a) Mapping Survey Procedure - A high-level view



(b) Classification Taxonomy

Fig. 2. Survey details: procedure and taxonomy.

Our approach uniquely integrates the base taxonomy with a dynamic analysis of emerging trends and challenges in control software engineering. This integration maps existing methodologies and critically examines their evolution and adaptability to changing industrial needs, thus addressing the gap between theoretical research and practical application.

An initial *base taxonomy* is developed based on our fundamental knowledge of the areas of control applications and software engineering. Elements are added in the *base taxonomy* during article search to form a *classification taxonomy*. Figure 2(b) shows the complete classification

taxonomy comprising the parent and leaf elements. The first two levels from the top represent the categories (*base taxonomy*), while the subsequent levels represent the sub-categories under which papers are bucketed. A "Base Taxonomy" is defined based on identifying all original articles. Figure 2(b) shows the taxonomy to categorize the articles in our study. The detailed description of the taxonomy is given below.

2.1 Classification Taxonomy

The classification taxonomy is organized into distinct categories that align with fundamental aspects of control software engineering. Each category is crucial for structuring the extensive data gathered from our literature review.

Meta-data Elements: These include:
—Title, authors, publish year, and journal/conference to aid in future categorization and analysis.

Engineering Phases: Articles are classified by software engineering phases discussed, including requirements, design, implementation, testing, and operations [46, 83].

Engineering Methods: Identifies methodologies like waterfall and agile, which represent traditional and modern approaches in software development, respectively. The decision to exclude the V-model was based on its lower prevalence in current literature [52], suggesting a shift toward more flexible methodologies.

Engineering Paradigms: Focuses on model-driven, code-driven, and knowledge-driven approaches, each offering different levels of abstraction and guidance in the software process [11, 41, 69, 115, 116, 128].

Target Application Domains: Describes domains like CPS, IoT, robotics, and automotive systems, highlighting how control software is tailored to meet specific operational needs within these domains [13, 28, 107, 144].

2.1.1 *Development and Continuous Update of Taxonomy.*

The development of the base taxonomy was methodically executed through a structured literature review, identifying key themes and categorizations within control software engineering. This taxonomy is continuously updated to reflect new research findings and technological advancements, ensuring its relevance and utility. The update process involves:

(1) Regular reviews of emerging literature to integrate new themes.
(2) Adaptations to accommodate new trends and feedback from users.

This approach ensures the taxonomy remains current and is an effective tool for organizing and analyzing the evolving field of control software engineering.

2.2 Utility and Continuous Evolution of the Taxonomy

The base taxonomy is continuously updated to incorporate new research findings, technologies, and trends, reflecting current state-of-the-art control software engineering. This dynamic update process involves regularly reviewing and integrating relevant new literature into the taxonomy, thus maintaining its relevance and utility in categorizing and analyzing emerging research areas.

The study is scoped by limiting the "Target Application Domains" to *CPS, IoT, robotics*, and *Automotive systems*. These four elements cover most application domains and are also identified as the driving technologies for Industry 4.0 [8]. RQs are derived and answered based on the taxonomy with the study results. Our choice of CPS, IoT, robotics, and automotive systems as target application domains is predicated on their prominence as primary fields where control software plays a crucial role [64, 75, 106, 123].

We recognize that robotics and automotive systems can be considered sub-categories of CPS and that there are overlaps between CPS and IoT. However, we classify them as separate target domains to highlight each field's unique challenges and advancements. This distinction allows us to provide a more detailed analysis of control software applications and their impact across various contexts.

These domains collectively represent a broad spectrum of modern engineering challenges and advancements. CPS and IoT encompass the integration of networked digital elements with physical processes, which are central to today's industrial and consumer applications. Robotics and automotive systems, both subsets of CPS, are at the forefront of applying sophisticated control software for autonomous and highly interactive tasks in manufacturing, healthcare, transportation, and service sectors [94]. This focus ensures our research covers the most impactful and transformative areas of control software application, reflecting significant trends in the evolution of interconnected, intelligent systems.

To ensure the completeness of our taxonomy, we designed each classification to cover all pertinent aspects of control software engineering from other mapping studies [12, 28, 37, 70, 92, 148]. This thorough approach ensures that every potential category of interest within the research community is represented, facilitating comprehensive analysis and synthesis of the literature. By categorically mapping out distinct areas such as engineering phases, methods, paradigms, and target application domains, our taxonomy not only aids in systematically classifying current research but also highlights under-represented regions that may benefit from further investigation. This level of detail supports the study's aim to bridge concepts and practical applications and provides a foundation for effectively identifying and addressing research gaps.

This base taxonomy is not just a classification tool but a bridge connecting theoretical concepts with practical challenges in the field. By continuously updating the taxonomy with recent literature, we ensure that it reflects the current state-of-the-art and addresses the real-world demands of control software engineering.

2.3 RQs

The various elements identified in the base taxonomy populate the RQs. Our study carefully selected 11 RQs derived from our base taxonomy, focusing on their critical importance to advancements in Industry 4.0. These questions emphasize key aspects such as automation, cost-effectiveness, ease of development, and the safety and security of systems in manufacturing technologies [8, 48].

The selected questions address crucial elements in control software engineering, including integrating agile practices, applying **Model-Driven Approaches (MDAs)**, and developing advanced security measures. This focus is particularly relevant due to modern industrial systems' increasing complexity and integration demands. The importance of robust, adaptable, and secure control software is underscored by recent literature, which highlights the significant technological and operational challenges facing Industry 4.0 [70, 125].

Our research aims to provide academically rigorous insights and produce findings highly applicable to contemporary industrial practices. This approach helps bridge the theoretical and practical aspects of Industry 4.0, ensuring that our study contributes effectively to the field. To refine our focus, we consulted with industry experts and professionals specializing in control software development for Industry 4.0, who helped assess each question's relevance and potential impact. This expert input was instrumental in prioritizing the most critical questions for advancing the field and addressing the challenges identified. We did not select specific questions, such as those focusing exclusively on niche applications, theoretical models without practical validation, or areas with limited impact on Industry 4.0 advancements. These questions, while valuable, were deemed less critical for our study's objectives, which aim to address broad, impactful, and contemporary issues in control software engineering.

We intentionally excluded questions focused exclusively on niche applications, theoretical models without practical validation, or areas with limited impact on Industry 4.0 advancements. For example, questions such as "What are the specific challenges in control software for agricultural automation?" and "What are the theoretical models proposed for control software in quantum computing systems?" were excluded due to their narrow focus and limited immediate applicability to broader industrial contexts. Similarly, questions addressing precise technical details, like "What are the specific implementation challenges of control software in real-time operating systems?" and emerging but less documented areas, such as "How effective are control software solutions in managing energy consumption in smart grids?" were not included to maintain the study's focus on widely applicable and practically relevant issues. The overall effect of not including these questions is that our research remains concentrated on the most impactful and broadly pertinent issues of control software engineering, ensuring that our findings provide significant and actionable insights for advancing Industry 4.0 practices:

(1) *RQ1: How actively is research conducted in control software engineering?* This meta-RQ examines the growth and decline in the popularity of control software engineering research.
(2) *RQ2: The research approaches support which engineering phases?* This question seeks to identify solutions to challenges in various control software engineering phases.
    (a) *RQ2.1: Which research solutions address multiple engineering phases?* We identify approaches that provide support across multiple engineering phases or are constrained to a single phase.
(3) *RQ3: Which engineering methods are predominant in constructing control software?* This question explores the applicability of waterfall, spiral, and agile practices in control software engineering.
(4) *RQ4: Which engineering paradigms does control software engineering rely on?* This question examines different levels of abstraction in software engineering, such as code-driven, model-driven, and knowledge-driven approaches [5, 24].
    (a) *RQ4.1: Which engineering phase is supported by which paradigm?* We identify the association between paradigms and their respective engineering phases.
(5) *RQ5: What artifact type is proposed to address control software engineering problems?* This question categorizes research contributions into tools, concepts, programming frameworks, languages, formalisms, or architectures.
    (a) *RQ5.1: Which artifact type suits which engineering phase?* We examine the applicability of different artifact types to specific engineering phases.
(6) *RQ6: Which application domains are most studied?* This question identifies the domains that use control software, such as CPS, robotics, IoT, and automotive systems [138].
    (a) *RQ6.1: Which approaches have multiple target application domains?* We determine if research provides generic approaches applicable to multiple domains.
    (b) *RQ6.2: Which engineering phases are supported by the approaches for which target application domains?* We explore the association between research approaches, application domains, and engineering phases.
(7) *RQ7: Which concerns are addressed by the research articles, using which paradigms and in which domains?* This question maps concerns such as safety, security, reliability, domain awareness, data manipulation, and performance to their respective paradigms and domains.

Our RQs are designed to explore theoretical foundations and uncover practical implications and applications of control software engineering in various industrial domains.

Table 1. Base Search Strings

Sr. No	Base Search Strings
1	(*"control software engineering"* **OR** *"robot software engineering"* **OR** *"cyber-physical system software engineering"* **OR** *"internet of things software engineering"*) **AND** ((*"engineering"* **OR** *"development"* **OR** *"designing"*) **AND** (*"environment"* **OR** *"platform"* **OR** *"approaches"* **OR** *"tools"*) **AND** (*"robot"* **OR** *"internet of things"* **OR** *"cyber-physical system"* **OR** *"control"* **OR** *"distributed"* **OR** *"control systems"*))
2	("control software" **OR** "control systems software" **OR** *"cyber-physical software"* **OR** *"distributed systems software"*) **OR** *"robot software"* **OR** *"orchestration software"* **OR** *"control and actuation software"* **OR** *"embedded software"***AND** ((*"engineering"* **OR** *"approach"* **OR** *"platform"* **OR** *"environment"* **OR** *"tool"*) **OR** (*"validation"* **OR** *"design"* **OR** *"requirements"*))

2.4 Sources Selection and Keywords

This survey is conducted on standard digital libraries. The list of libraries used for the searches are: (1) Science Direct, (2) ACM Digital Library, and (3) IEEE Library To further cross-verify and evaluate the completeness of our search results, search engines like "Google Scholar" and "Semantic Scholar" are also considered. The keywords that guide the search with the logical connectors, "AND" and "OR," are shown in Table 1. The query string is customized for every search engine to match their respective format.

To develop our search strings, we extracted key concepts from our established taxonomy, categorizing the primary areas of interest in control software engineering: "Cyber-Physical Systems," "Robotics," "Internet of Things," and "Automotive Systems." We analyzed each category's academic literature and industry discourse to identify frequently occurring and relevant keywords. The search strings were constructed using these keywords and logical connectors "AND"/"OR." This methodology allowed us to comprehensively capture articles across these categories, ensuring an exhaustive and targeted search. The connectors helped refine the search to include papers that discuss these keywords in various combinations, covering multiple aspects of control software engineering. The resulting search strings, shown in Table 1, were optimized to yield the most relevant results for our study's focus areas.

2.5 Search Process

Search queries are formulated to gather primary studies that are further used to identify more articles by analyzing the cited references in the respective papers. Every paper's title, keywords, and abstract are examined for relevance. In some instances, where more clarity was needed, the introduction and approach sections of the paper were also considered. We select the studies dealing with process improvement, quality, coordination, and related issues in control software engineering based on title, abstract, and keywords. Sometimes, it was necessary to read the entire document to determine its relevance.

2.6 Inclusion and Exclusion Criteria and Quality Assessment

The selection of articles was systematically conducted to ensure comprehensive coverage of the field of control software engineering. The process was iterative, involving multiple stages to refine and validate the search results.

(1) *Initial Broad Search:* We initiated our study with a broad search using generic keywords relevant to control software engineering across major databases, including Science Direct, ACM Digital Library, IEEE Library, Google Scholar, and Semantic Scholar. This initial stage aimed to collect as many potentially relevant articles as possible without filters on publication date or document type. However, we limited our search to articles published up until 2020. This strategic decision focuses on well-established research with sufficient time to demonstrate impact and accrue citations, critical indicators of the work's influence within the field. This approach ensures a comprehensive perspective on trends while avoiding uncertainties associated with the latest, potentially unvetted publications.
(2) *Screening of Titles and Abstracts:* All retrieved articles were screened based on titles and abstracts. The primary focus was identifying papers explicitly discussing software engineering approaches, methods, and challenges specific to control systems in identified application domains like CPS, IoT, robotics, and automotive systems.
(3) *Full-Text Review:* Articles that passed the initial screening underwent a full-text review to assess the depth of discussion regarding control software engineering. This review helped understand whether the articles proposed solutions, addressed significant engineering phases, or presented validated results.
(4) *Exclusion of Irrelevant and Redundant Studies:* During this phase, we excluded studies that did not directly contribute to the understanding or developing of control software systems. Specifically, studies that were:
    —Purely theoretical without direct applications or implications for control software.
    —Not applicable to the targeted domains of CPS, IoT, robotics, or automotive systems, such as those focusing exclusively on unrelated fields.
    —Primarily concerned with hardware aspects without significant software engineering content.

    Redundancies were also addressed by selecting the most comprehensive or recent publications when multiple reports from the same research initiative were available.
(5) *Application of Inclusion Criteria:* Articles were included based on the following specific criteria to ensure relevance and quality:
    —We defined "full papers" as those with at least eight pages to ensure sufficient detail for substantive analysis. This threshold ensures the inclusion of literature review, methodology, analysis, and conclusions, aligning with standards in software engineering where longer papers indicate thorough research and detailed results.
    —Were peer-reviewed, ensuring the quality and credibility of the research.
    —Offered original research with clear, actionable solutions, not merely identifying challenges.
    —Specifically discussed aspects of software engineering within CPS, IoT, robotics, or automotive systems.
(6) *Quality Assessment:* Each selected article was assessed for quality based on:
    QA1: Peer-reviewed status, confirming the research underwent rigorous evaluation.
    QA2: Validity of the approaches used, ensuring that methods are sound and justifiable.
    QA3: Contributions to control software engineering, such as innovations that reduce time, effort, and cost, enhance modeling techniques, or introduce novel architectures or programming frameworks.

The methodology was designed to be iterative and dynamic, allowing adjustments based on the evolving understanding of the field. This approach ensured the selection process was systematic

Table 2. Sources Distribution

Sources	Found	Unique	Relevant	Primaries
Science Direct	216	77	62	39
ACM Digital Library	364	211	81	34
IEEE Digital Library	423	354	76	42

and comprehensive, capturing the most relevant and high-quality papers contributing significantly to control software engineering.

3 Categorization and Analysis of Bibliographic Data

After selection, bibliographic data for each article were meticulously categorized using a specially developed taxonomy designed to encompass the full scope of control software engineering. This process involved several key steps:

(1) *Development of the Taxonomy:* We initially established a base taxonomy, outlining key categories and sub-categories relevant to the field, based on our preliminary literature review. This was regularly updated to include emerging trends and ensure accuracy and comprehensiveness.
(2) *Assignment of Articles to Categories:* Articles were assigned to specific categories within the taxonomy, such as engineering phases (requirements, design) and application domains (CPS, IoT, robotics, automotive), based on their primary focus and contributions.
(3) *Dynamic Updating of the Taxonomy:* The taxonomy was dynamically refined as categorization progressed, integrating new insights and adapting to changes in the field to maintain relevance.
(4) *Validation of Categorization:* Team members rigorously reviewed each categorization to ensure accuracy. Discrepancies were resolved through consensus, enhancing the taxonomy's reliability.
(5) *Statistical Analysis and Visualization:* Using R, we performed statistical analyses and created visualizations to identify trends, gaps, and research clusters. These visual tools helped illustrate the research distribution across different categories, providing insights into the landscape of the field.

This structured approach ensured that our analysis was thorough and the findings well-supported by a robust methodological framework. By continually updating and validating the taxonomy, we maintained a flexible yet reliable structure for organizing extensive data, enabling us to draw meaningful conclusions about the state and future directions of control software engineering.

4 Results

The search procedure produced 567 original articles, of which 329 were unique. One hundred seventy of these were relevant to the study, and 115 articles were identified as primary studies (refer to the Appendix for the detailed list of the prior studies). Table 2 shows the distribution of studies found according to the sources used.

The results provide a comprehensive overview of the current state of control software engineering and highlight specific areas where practical applications lag behind theoretical advancements. This gap, evident in the trends and distributions of the research, underscores the need for more focused and application-oriented research in the field.

4.1 Data Extraction and Analysis

R Environment [71] converts the bibliography file into a data frame for statistical analysis. The articles in the bibliography are scored by assigning "+1" for every category an article falls in and "0" for all the remaining classes from the taxonomy. The summation of all the scores against each article denotes a final impact score of the article. This process results in a table with the article, categories, and last impact score columns for the table. The results data are analyzed to answer the RQs in Section 2.3.

The impact score analysis reflects each article's academic relevance and indicates their potential practical applicability. Articles with higher impact scores often present more comprehensive approaches, which could be more directly applicable in real-world control software engineering scenarios.

We selected the R platform for its extensive packages and robust support for data manipulation, making it ideal for analyzing our complex datasets. R's powerful statistical capabilities and flexibility meet our study's analytical demands, while its open source nature ensures transparency and reproducibility. We used UpSetR [30] for its ability to visualize complex intersections of sets, aiding in analyzing overlaps within our taxonomy. Additionally, the ComplexHeatmap [53] package from Bioconductor [55] was employed to create detailed heatmaps, essential for displaying multiple layers of information and facilitating a nuanced interpretation of data relationships.

These tools were selected based on their technical merits, broad acceptance, and ongoing support within the academic community, ensuring their reliability and relevance. This choice was specifically tailored to the needs of our analysis, which involves integrating and visualizing intricate bibliographic data to address our RQs comprehensively. The detailed categorization of the extracted data is given in Tables A1, A2, A3, A4, A5 and A6.

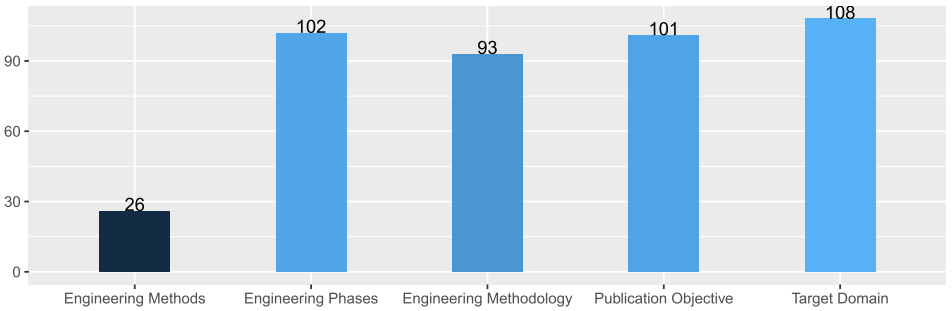
4.2 Trends in Publication Areas

Figure 3(a) shows the distribution of articles per category, while Figure 3(b) indicates the number of articles published against the year and the score distribution. From this plot, it can be inferred that articles under *Engineering Methods* are only 26, while other categories in the taxonomy have more than 90 articles each. There are comparatively fewer research articles on engineering methods in control software.

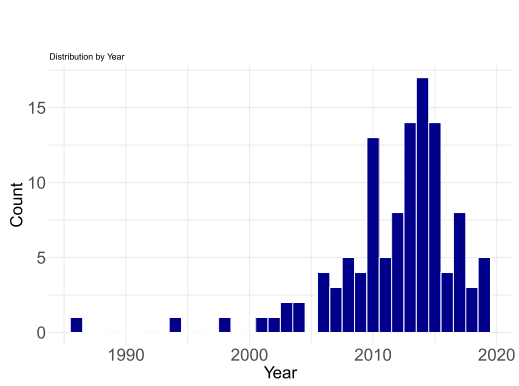
The year-wise distribution clearly shows that the number of articles on control software engineering gained significant popularity in early 2000. A point to notice here is that this is the same period when robotics, industrial automation, and IoT gained immense popularity [65]. This trend indicates that the rise in the number of articles on control software can be directly associated with the business value and popularity in the target application domain. However, after 2015, there seems to have been a sudden drop in the number of articles in this area. An interesting observation is that the industrial sector had laid the vision for Industry 4.0 during the same time. [125]. Based on the above statement, there is expected to be a significant rise in research on control software engineering when the industries adopt Industry 4.0.

The observed trends in publication areas offer valuable insights into the evolving needs and challenges of control software engineering in practice.

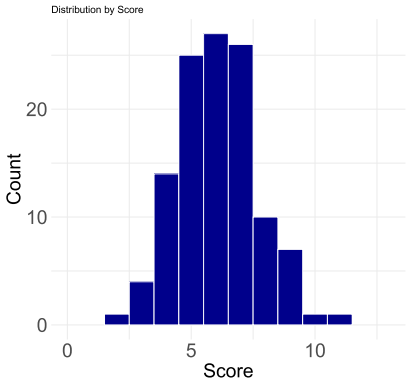
As discussed above, the scoring criteria allow us to identify an article that covers a larger space in our category taxonomy. As seen in the plot in the bottom right, the articles with a score range of 4–6 are in a more significant number. However, the total number of categories and sub-categories in our taxonomy is 21, which means the maximum scoring limit for an article is 21. However, the plot shows that our study has only one article scoring the highest, i.e., 10. Most of the articles fall in the range of 5–8, which shows that the articles from the literature target particular topics, and



(a) Categorisation of articles



(b) Publications per year and score distribution



(c) Score distribution

Fig. 3. Trends in control software engineering.

very few articles cover multiple engineering aspects, paradigms, concerns, and target domains. Few articles propose approaches that provide an end-to-end engineering approach in control software.

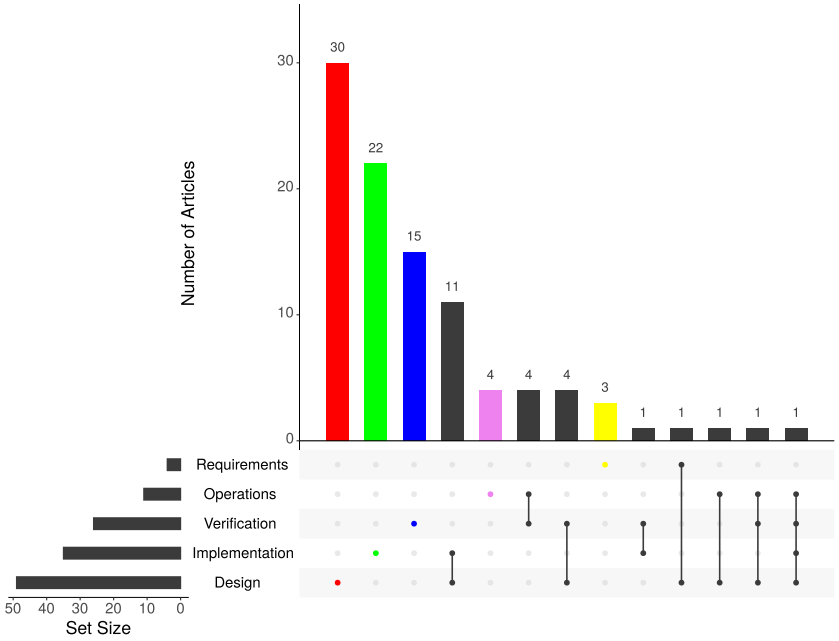
The analysis above answers our first RQ, RQ1: How actively is research being conducted control software engineering?

(1) The trend in research articles in control software engineering seems to rise when there is a demand in the industrial and business sectors. That is why there was a rise in the articles around 2000, when CPS, robotics, and IoT showed industrial applications. Research articles are expected in this area as industry and business sectors rapidly adopt Industry 4.0.
(2) *Engineering Methods* for control software has the lowest number of articles, while the other categories have many. This observation shows the trend in choosing the topics of research.

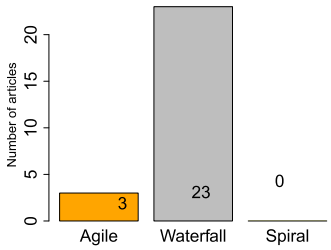
4.3 Engineering Phases Supported by the Research Publications

Of the 115 identified primaries, 102 articles define approaches that support one or many *engineering phases* as described in the taxonomy. Ten articles define approaches which support both *design* and *implementation*, while only one article covers the areas of *design*, *implementation*, *verification*, and *operations*.

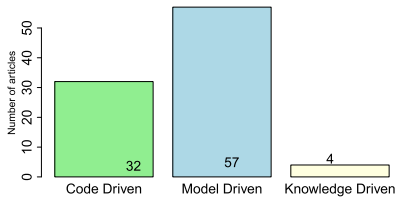
The plots in Figure 4(b) and (c) show the distribution of engineering methodologies (*agile, spiral*, and *waterfall*) and approaches (*code-driven, model-driven,* and *knowledge-driven*). It can be inferred that the approaches in the literature prefer *Waterfall* for control software engineering. From Figure 4(c), it can be seen that the *MDAs* are the most popular techniques. However, upcoming



(a) Literature distribution for Engineering Phases



(b) Distribution of Engineering Methodologies



(c) Distribution of Engineering Approaches

Fig. 4. Trends in control software engineering.

methods like *agile* and *knowledge-driven* approaches are also seen in small numbers, slowly gaining popularity in the modern engineering community. Based on the above observations, the RQs (1) RQ2: The research approaches support which engineering phases? (2) RQ3: Which engineering methods are predominant in constructing control software? and (3) RQ4: Which engineering paradigms do control software engineering rely on? are answered:

(1) Most approach targets specific engineering phases, making these approaches one-off.
(2) A small number of approaches cut across multiple engineering phases.
(3) Eleven approaches are seen to cut across *design* and *implementation* phases and only one approach cuts across *design-implementation-verification-operation* phases.
(4) Waterfall methodology is predominantly used in control software engineering, while only four approaches support agile.
(5) *MDA* is the most dominant approach in control software engineering, while *knowledge-driven approaches* exist on a smaller scale.

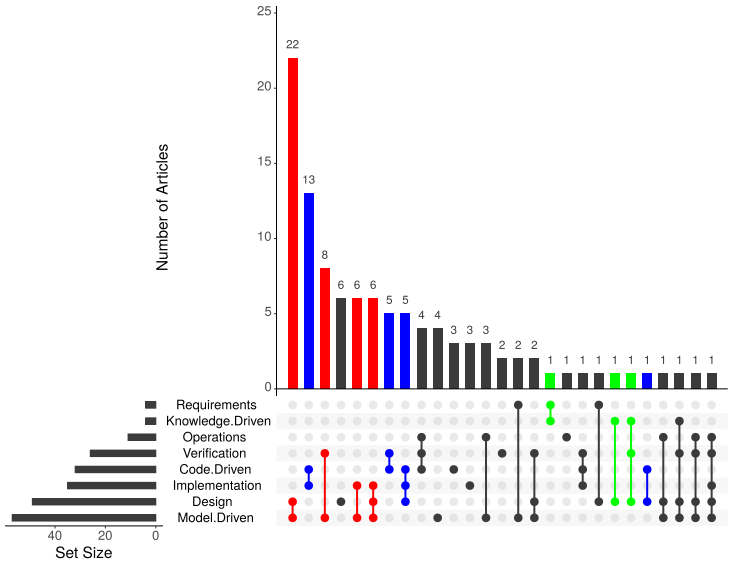


Fig. 5. Publications supporting specific engineering phases by using different engineering paradigms.

(6) The spiral methodology, though part of our classification, was not adopted in any identified studies, indicating its limited use in this context. Although not adopted by any identified studies, the spiral model is included in our classification to ensure completeness and acknowledge its theoretical significance in software development. Its iterative and risk-focused approach is relevant to control software engineering. It highlights a gap in current research and presents an opportunity for future exploration. This inclusion aligns our study with established software engineering frameworks, enhancing the comprehensiveness and rigor of our analysis.

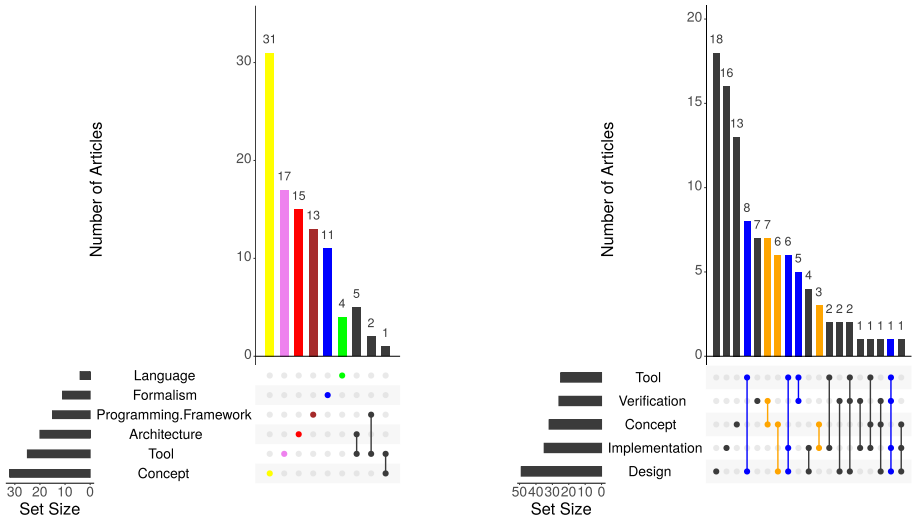
The distribution of methodologies and approaches across different engineering phases indicates a disparity in research focus, with certain phases receiving more attention than others. This disparity has practical implications, as it may lead to underdeveloped methodologies in less researched phases, affecting the overall efficacy of control software engineering in the industry.

4.4 Engineering Paradigms Used in Research Publications

Figure 5 illustrates the distribution of articles across different engineering paradigms, highlighting how each approach supports various phases of software engineering. In the figure, black bars represent *code-driven* approaches, red bars signify *MDAs*, and green bars denote *knowledge-driven* approaches.

MDAs are predominant in the design phase, reflecting their structured nature ideal for environments requiring precise control and predictable outcomes, such as automotive or aerospace industries. Their structured methodology supports upfront planning and rigorous documentation, aligning well with traditional engineering practices and regulatory frameworks.

Code-driven approaches offer flexibility, making them suitable for rapidly evolving project environments where adaptability is paramount. These approaches are primarily utilized in the implementation phases. They are less prevalent in the literature that emphasizes structured methodologies, indicating their preference in industries valuing agility, such as startups or tech companies using agile methods.



(a) Research Objectives Distribution (b) Tool vs Concept for Engineering Phases

Fig. 6. Research objectives of publications and their relation with engineering phases.

Knowledge-driven approaches are less represented but are increasingly recognized for their utility in environments that require adaptive decision-making capabilities, like autonomous robotics or systems incorporating AI and machine learning. These approaches are crucial for developing software that evolves based on real-time data, highlighting their potential in advancing autonomous system technologies.

- —*MDAs* are favored in design and test planning phases, underscoring their utility in environments where detailed planning and verification are crucial.
- —*Code-driven approaches* excel in phases requiring rapid modifications, such as implementation and maintenance.
- —*Knowledge-driven approaches* are instrumental in integrating cutting-edge technologies, making them invaluable for projects in dynamic, evolving environments.

This analysis reveals that while MDAs continue to dominate due to their alignment with well-established engineering practices, *code-driven* and *knowledge-driven* methodologies are gaining traction in scenarios demanding greater flexibility and adaptability. This trend reflects the shifting priorities in control software engineering as industries move toward more dynamic and autonomous systems.

4.5 Research Objectives of the Publications

Figure 6 answers RQ5: What is the artifact type proposed to address problems in control software engineering? Most research artifacts are presented as theoretical and formal concepts, while very few are transformed into usable tools or products. We also observe that the tooling support from the research articles exists for the design and implementation phases.

4.6 Target Application Domains Supported by the Research Publications

As discussed in the previous sections, CPS, the IoT, robotics/mechatronics, and automotive have been chosen as target domains for our study. The articles are also analyzed to understand the

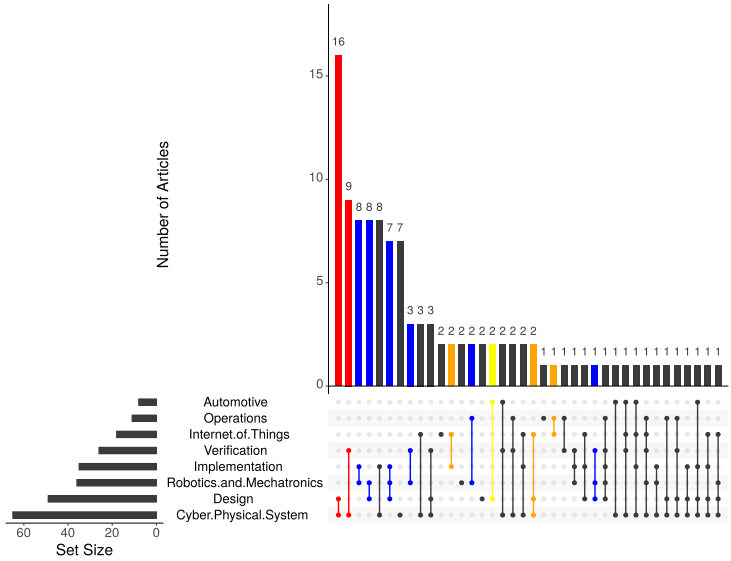


Fig. 7. Publications supporting a specific engineering phase for a specific target application domain.

links between the target domains, engineering approaches, and methods. This insight gives us a high-level understanding of how engineering approaches get applied in these domains.

The black highlighted bars in the plot represent the articles for the *robotics* domain and their corresponding engineering phases. Similarly, red, orange, and yellow highlighted bars correspond to the *CPS, IoT*, and *automotive* systems. Figure 7 shows that most of the *design* approach in the literature applies to *CPS*, followed by *design* and *implementation* for the *robotics* domain. Most of these approaches are used in the *robotics* domain (black), while several other approaches seem shared across *CPS* and *IoT* (eight articles). *Automotive* systems find a place in this list having articles proposing design approaches.

Based on this understanding, RQ6: Which application domains are most studied? is answered.

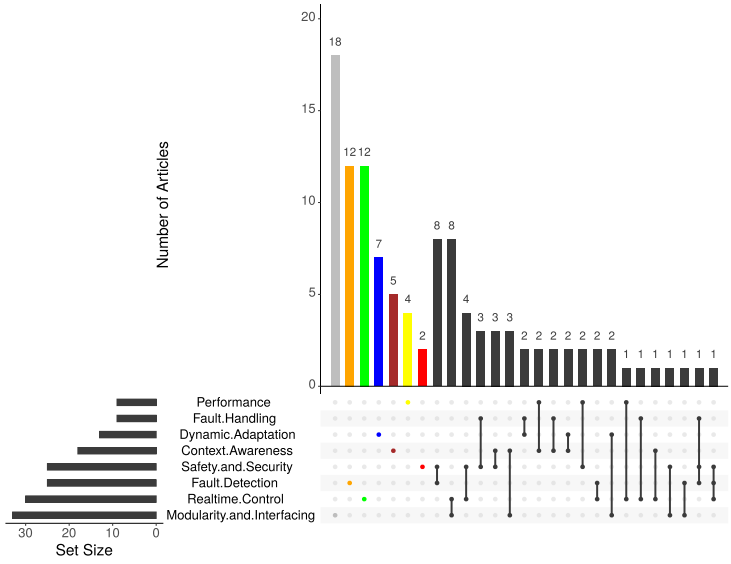
(1) Most of the design and implementation approaches apply to cyber-physical and robotic systems
(2) Most of the testing approaches apply to CPS.
(3) Very few approaches are conventional across all the target domains, and no standard approach applies to all the target domains.

4.7 Concerns Addressed in the Research Publications

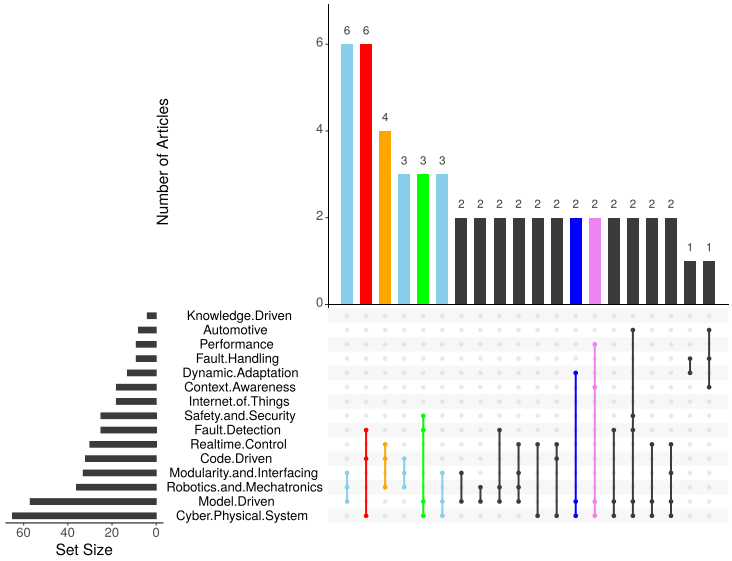
Figure 8(a) shows the major concerns handled by the control software. Figure 8(b) offers the originating target domains for the concerns and the engineering paradigms used to address them.

As seen in the above figures, the major concerns (RQ7: Which concerns are addressed by the research articles, using which paradigms and in which domains?) that are identified in this study are

—*Modularity and Interfacing*: The control software enables the hardware/software modules to interface and interact. This concern is addressed in 16 articles and is the top concern in the study.



(a) Identified concerns for control software



(b) Target application domains, their concerns and engineering paradigms

Fig. 8. Research concerns.

—*Realtime Control*: Control software controls the underlying sub-systems in real-time and can be considered a critical concern. This assumption receives backing from the study as there are 12 articles which specifically address the concern of *Realtime Control*.
—*Fault Detection*: Detection of faults in the system has 12 articles addressing this concern.

Apart from the above concerns, this study also identifies concerns like *Dynamic Adaptation* (seven articles) and *Context Awareness* (five articles).

Figure 8(b) shows the correlation between the concerns, their corresponding target domains, and the engineering paradigms used to address them. Based on the data shown in the figure, the following observations are made to answer the seventh RQ:

- —The MDA is mostly used to address the concern of *modularity and interfacing* in *robotic and mechatronic* systems.
- —The concern of *fault detection* in CPS is addressed using the *code-driven approach*.
- —*Code-driven approach* is also used to achieve *realtime control* in *robotic and mechatronic* systems.
- —*MDA* is seen to be used to address two concerns, *safety and security* and *fault detection* in *CPS*.

As it can be seen, no approach is utilized across multiple target domains to address numerous concerns. The reason is domain dependency; most articles focus on a specific domain.

4.8 Control Software Engineering Map

This study focuses on mapping the different approaches, engineering methods, target technology domains, and phases for control software engineering. A systematic mapping approach was followed to identify 115 studies as primaries. This study categorizes data to determine the contributions in the studies, solution approaches, engineering methods, and target technology domains. The studies and the categorization are analyzed to identify and establish a correlation between them. Based on the correlation, the RQs are answered.

A control software engineering map is shown in Figure 9 as a categorized-split-heatmap [54]. Every split represents the intersection of the categories on the *X*- and *Y*-axes, e.g., the top-left split map represents the intersection of *Engineering Methods*, on both *X*- and *Y*-axes. The color of every cell and the number in it represents the number of interventions corresponding to the intersection of elements along the *X*- and *Y*-axes. With this representation, it is seen that the splits which have a higher density of darker shades like *Engineering Phases* ∩ *Target Application Domains* have more extensive interventions as compared to the more significant density of lighter shades like *Research Outcomes* ∩ *Research Concerns*.

This control software engineering map is a theoretical framework and a practical tool for identifying and addressing gaps in research and practice. Highlighting the intersections and density of research in various categories provides a clear direction for future research and development efforts in control software engineering. In summary, our study results map the current landscape of control software engineering research and serve as a guide for aligning future research efforts with practical industry needs. This alignment is crucial for advancing the field in an academically robust and practically relevant direction.

5 Discussions

This section discusses the findings from this study based on the data gathered from the articles and the analysis. This section also points out the gaps based on the study of the primary articles, based on which potential research areas are proposed.

Control Software Engineering—Requirements Phase. While most of the research targets engineering design, implementation, and testing phases, the requirements and operation phases still have less research in control software. As requirements engineering is an immensely evolved area in software engineering research, these learnings could be used to control software requirements. Unlike traditional software engineering, sufficient support for the requirements phase was not found in formal/informal requirements specification languages, requirements modeling frameworks, and

Fig. 9. Control software engineering map.

platforms. This creates an opportunity to explore and develop research artifacts to enable a formal approach to capture requirements for control software engineering projects.

Control Software Engineering—Design Phase. Model-driven methods have been extensively used to develop design specification languages and implement composition and synthesis-based approaches. Knowledge-driven strategies could also be a potential contender to aid MDAs for contextual adaptation. In this study, articles like KnowRob [129] and RoboEarth [109] have proposed knowledge-driven approaches to synthesize robotic task-recipe and behavior based on changing context. Hence, research on the applicability of knowledge-driven approaches to modeling controller design and behavior could be suited to future systems engineering.

Control Software Engineering—Implementation Phase. State-of-the-art software frameworks, libraries, and code generators support the implementation phases. It is observed that a system's implementation of control software is highly dependent on its domain. The research community

should examine how domain-specific variations could be coupled with the implementation technologies to enable precise implementation.

Control Software Engineering—Verification Phase. Our study shows that verification strategies are one-off and specific to a target domain and underlying hardware. However, these approaches are difficult to reuse for other domains. State-of-the-art provides test coverage and test case generation at the level of the components in almost all the techniques. While the available methods provide simulation platforms for testing, the configuration of these platforms is still manual, requiring domain expertise, hence adding to the cost. As these configurations can be the base knowledge to set up the test environment, the opportunity exists to explore knowledge-driven approaches to store testing knowledge for a specific domain and utilize it to populate the test configurations.

Control Software Engineering—Operations Phase. In this study, the operation phase is found to have fewer articles than the other engineering phases. As a system evolves, its control software also evolves, leading to design, code, and operation changes. A system's operations validate whether the system achieves its desired goal. Few articles address this phase, and the approaches used in these articles are one-off and exhibit limited reuse across multiple domains.

End-to-End Control Software Development Practices. As seen in Figure 4(a), the absence of systematic end-to-end engineering solutions in the literature reveals significant challenges, such as the incompatibility among specialized modeling tools and the unique requirements of diverse target domains, which impede universal engineering practices. Opportunities exist in developing domain-specific modeling languages [102, 113] to enhance tool integration and process streamlining. Additionally, establishing domain-specific and knowledge-driven methodologies [6, 11, 26] could align specific domain needs with broad engineering practices. Furthermore, standardization initiatives could enhance tool compatibility and interoperability across various phases and domains, promoting more effective engineering practices.

Engineering Paradigms. Models are easy to create and understand but challenging to maintain and update; also, they become obsolete. Knowledge-driven methods like KnowRob [129] and RoboEarth [73, 109] provide a new perspective to enable the sharing of information in the form of ontologies rather than models. This also opens opportunities to integrate knowledge-driven approach and MDAs to build future systems.

Engineering Methods. Engineering methods play a vital role in the time and cost of the engineering process. Having an advantage over traditional approaches, Agility must find a suitable place in control software engineering. Only one agile approach in our study provides a platform-based tool for control software engineering. Going ahead, the need for agile-based approaches to engineer control software, which can be used across multiple application domains, could be experienced.

Control Software Engineering—Tool Support. Many approaches from the articles propose tooling support for specific engineering phases. However, these tools are particular to the target domains, programming language, or engineering phase. Many of these tools are challenging to use in conjunction with each other, restricting the creation of toolchains based on the demands of the application domain.

Approaches Specific to Application Domain. Domains like CPS, robotics, IoT, and automotive systems significantly influence control software engineering research areas. Many studied articles were specific to the target domain [36]. In some instances (e.g., [130]), approaches were seen to be quite similar from the control perspective but very tightly coupled to their application domain. The specific nature of such procedures restricts reuse. We see the need for a domain-driven engineering approach that can be customized for a target domain.

Applicability of Engineering Paradigms to Diverse Application Domains. Very few articles propose a systematic end-to-end (requirements-design-implementation-testing-operations) engineering approach. As these approaches exist for specific target systems, it becomes difficult to link them. This lack of support is mainly due to the incompatibility of models and tools with each other and diverse target domains. Another opportunity is to explore a common vocabulary to describe the control specifications irrespective of the target domain. As seen from the study, few articles have suggested domain-independent approaches, but implementing these approaches always favors a specific domain (e.g., ROS [105]).

Target Application Domains. All the target application domains like robotics, IoT, and CPS influence the approaches in the articles. The articles suggest novel approaches to engineering these systems, but simultaneously, the methods are particular to the target domain. As seen in the previous paragraphs, this specificity restricts the reuse of these approaches. The methods we studied do not have a common base origin, like a formal control model or theory. Hence, there is a strong need to invest more in exploring the nature of domains and how they affect the control software in various ways.

Addressing Engineering Concerns. Handling concerns is specific to a domain; hence, the approaches are not reusable across multiple domains. While working on this study, it was realized that there exist interdependent concerns like *fault-detection and fault-handling*, *dynamic adaptability and context awareness*, *performance and realtime control*, which are interdependent. For example, fault-handling techniques depend on fault detection. It was found that each of such interdependent concerns is handled individually in isolation from its dependent concern. The challenge in addressing interdependent concerns is the lack of approaches that can reason about the effect of addressing one concern on another.

The findings underscore the need for a more holistic approach to control software engineering encompassing all development phases. Future research should focus on creating adaptable methodologies and tools across various domains, reducing the domain-specific limitations currently observed. The study also points to the necessity of developing integrated approaches that simultaneously address interdependent concerns. Such approaches would allow for a more comprehensive understanding of the interplay between different aspects of control software engineering, leading to more robust and practical solutions.

6 Threats to Validity

In this SMS, we have identified several threats to validity based on the guidelines and classifications discussed by Wohlin et al. [142] in their work on experimentation in software engineering. These threats are categorized as follows:

(1) *Construct Validity: Taxonomy Development and Categorization:* The initial taxonomy and subsequent categorization of literature were based on our preliminary understanding, potentially omitting emerging or less-documented concepts. Despite continuous refinement to include new insights and reduce bias toward initial assumptions, some biases may persist.
(2) *Internal Validity: Selection of RQs:* The focus on 11 RQs, selected due to their relevance to Industry 4.0 and the prevalence of specific challenges, might have excluded other pertinent questions. This prioritization means some areas within control software engineering may remain unexplored in this study.
(3) *External Validity: Literature Search Constraints:* Our literature search was limited to specific databases using selected search terms, which might have missed relevant studies not indexed in these databases or not matching the chosen keywords.

(4) *Conclusion Validity: Screening and Selection Bias:* The screening process, though systematic, involved subjective judgments during title, abstract, and full-text reviews. Multiple reviewers were involved to minimize bias, with discrepancies resolved through discussion and consensus. However, the application of selection criteria might vary slightly among reviewers.
(5) *Reliability: Temporal Limitations:* The conclusions of this study are based on the literature available up to the review time. Given the rapid developments in control software engineering, newer studies post-review might not have been included, potentially affecting the study's relevance over time.

6.1 Internal Validity

Internal validity concerns whether the observed effects in a study are due to the treatment or other factors. We have identified the following potential threats:

— *Confounding Variables:* Factors other than the independent variable might influence the outcomes. We employed inclusion and exclusion criteria, applied statistical controls, and performed sensitivity analyses to control for potential confounders.
— *Selection Bias:* The process of selecting studies might introduce bias. We used a comprehensive and systematic selection process with multiple reviewers to ensure consistency.
— *Instrumentation:* Changes in measurement tools or procedures can affect results. We standardized our data extraction and analysis procedures.
— *Testing:* Repeated testing can influence outcomes. We performed independent data extraction and analysis.
— *History:* Events occurring between data collection phases might affect the study. We conducted the study within a specific timeframe to limit external influences.
— *Maturation:* Changes in the field over time can impact outcomes. Our study's timeframe is limited to publications up to 2020 to ensure consistency.
— *Regression to the Mean:* Extreme findings might regress to the mean. We included a broad range of studies to mitigate this effect.

6.2 External Validity

External validity relates to the generalizability of our findings beyond the specific conditions of the study. We have considered the following threats:

— *Population Validity:* The extent to which our findings can be generalized to the broader population. We selected a diverse range of studies to enhance generalizability.
— *Ecological Validity:* The extent to which our findings apply to real-world settings. We included studies from various application domains to reflect practical relevance.
— *Temporal Validity:* The generalizability of findings over time. Our study is based on literature up to 2020, and we plan to update it periodically to maintain temporal relevance.
— *Interaction Effects:* Interactions between the treatment and selection, setting, or time might influence results. We used diverse data sources, timelines, and classification categories to mitigate this threat.

6.3 Construct Validity

Construct validity concerns whether the study accurately measures the theoretical constructs it intends to measure. Identified threats include:

— *Operationalization:* We ensured rigorous operational definitions for our variables.

—*Mono-Operational Bias:* Using a single measurement approach might not capture the construct fully. We employed multiple data sources to get various results from articles for different domains and solutions to reduce this bias.
—*Mono-Method Bias:* We integrated various methodological approaches to ensure robustness.
—*Hypothesis Guessing:* Participants or authors might guess the study's purpose and alter their behavior. We maintained objectivity in data extraction and analysis. The reference taxonomy ensures that the data are always aligned with the objective.
—*Experimenter Expectancy:* We conducted blind reviews and analyses to prevent this bias. However, still being a highly relative aspect is very difficult to guarantee for this study.

6.4 Statistical Conclusion Validity

Statistical conclusion validity concerns using statistical tests to infer relationships between variables. Potential threats include:

—*Fishing and the Error Rate:* Conducting multiple statistical tests increases the chance of false positives. We used R studio (statistical platform) to make appropriate corrections for multiple comparisons.
—*Reliability of Measures:* The consistency of our measurement tools is critical. We used consistent data formats and tools to process the data.

By categorizing these threats, we aim to provide a structured understanding of the potential limitations of our study. Despite these challenges, the study's design—characterized by iterative updates, a comprehensive search strategy, and multi-level review processes—helps ensure a reliable and pertinent field assessment. We have strived for transparency in our methodology to facilitate replication and verification of our results, thereby strengthening the study's validity.

7 Conclusion and Future Research Directions

Based on the discussions above, it is evident that control software engineering is a dynamically evolving field with extensive research activities, especially in the design, implementation, and verification phases. MDAs are predominantly used, while knowledge-driven approaches are less common but show promise. This study underscores the need for innovative approaches that bridge the gap between theoretical research and practical applications. To advance the field, future research should focus on developing agile and flexible methodologies that can be adapted across various engineering phases and domains.

7.1 Proposed Future Research Initiatives

—*Integrated Toolchains:* Explore the development of integrated toolchains to enhance interoperability across modeling tools and platforms, addressing the challenges of tool and model incompatibility and facilitating smoother end-to-end engineering practices.
—*Standardization of Modeling Languages:* Aim to standardize domain-specific modeling languages to decrease fragmentation across engineering domains, fostering the development of universal or adaptable frameworks tailored for specific yet broadly applicable contexts.
—*Agile Methodologies:* Investigate the integration of agile methodologies in control software engineering, offering flexibility and quick responsiveness, which are particularly beneficial in fast-evolving sectors like IoT and consumer electronics.
—*Expansion of Knowledge-Driven Approaches:* Given their success in the design phase, extend knowledge-driven methodologies to other software engineering phases such as requirements, verification, and operations, enhancing their scope and utility.

— *Holistic Approaches to Interdependent Concerns:* Develop methodologies that cohesively address interdependent issues, such as fault detection and performance, potentially utilizing advanced AI and machine learning technologies.
— *Cross-Domain Methodologies:* Conduct studies to create adaptable, control-driven engineering approaches that can be applied across multiple domains, overcoming current limitations of domain-specific methods.

These research directions aim to bridge theoretical and practical gaps and enhance the agility and robustness of methodologies within control software engineering. We plan to benchmark effective approaches in specific areas and investigate interventions that could span multiple engineering phases and application domains. Additionally, we will explore how integrating agile practices into control software engineering could leverage adaptability and continuous improvement, essential characteristics of agile methodologies.

References

[1] Mohammad Abdullah Al Faruque and Fereidoun Ahourai. 2014. A model-based design of cyber-physical energy systems. In *Proceedings of the 2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC '14)*. IEEE, 97–104.

[2] Jonathan Aldrich, David Garlan, Christian Kaestner, Claire Le Goues, Anahita Mohseni-Kabir, Ivan Ruchkin, Selva Samuel, Bradley Schmerl, Christopher Steven Timperley, Manuela Veloso, et al. 2019. Model-based adaptation for robotics software. *IEEE Software* 36, 2 (2019), 83–90.

[3] Raian Ali, Fabiano Dalpiaz, and Paolo Giorgini. 2010. A goal-based framework for contextual requirements modeling and analysis. *Requirements Engineering* 15, 4 (2010), 439–458.

[4] Noriaki Ando, Takashi Suehiro, and Tetsuo Kotoku. 2008. A software platform for component based RT-system development: OpenRTM-aist. In *Proceedings of the International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 87–98.

[5] Gabriela Arévalo, Jean-Rémi Falleri, Marianne Huchard, and Clémentine Nebut. 2006. Building abstractions in class models: Formal concept analysis in a model-driven approach. In *Proceedings of the 9th International Conference on Model Driven Engineering Languages and Systems*. Springer, 513–527.

[6] Amar B., Subhrojyoti R. C., Barnali B., R. Dhakshinamoorthy, Seenivasan A., and Naveenkumar S. 2021. Knowledge driven rapid development of white box digital twins for industrial plant systems. In *Proceedings of the 47th Annual Conference of the IEEE Industrial Electronics Society (IECON '21)*. IEEE, 1–6. DOI: https://doi.org/10.1109/IECON48115.2021.9589912

[7] Bahareh Badban, Martin Fränzle, Jan Peleska, and Tino Teige. 2006. Test automation for hybrid systems. In *Proceedings of the 3rd International Workshop on Software Quality Assurance*. ACM, 14–21.

[8] Mohd Aiman Kamarul Bahrin, Mohd Fauzi Othman, N. H. Nor Azli, and Muhamad Farihin Talib. 2016. Industry 4.0: A review on industrial automation and robotic. *Jurnal Teknologi* 78, 6–13 (2016), 137–143.

[9] Malcolm Bain. 2014. Sentilo - Sensor and actuator platform for smart cities. Retrieved 20 February 2015 from https://interoperable-europe.ec.europa.eu/collection/egovernment/document/sentilo-sensor-and-actuator-platform-smart-cities

[10] Tucker Balch. 2002. The TeamBots environment for multi-robot systems development. In *Working Notes of Tutorial on Mobile Robot Programming Paradigms (ICRA '02)*, 1–16. Retrieved from https://www.cs.jhu.edu/~hager/Public/ICRAtutorial/Balch-Teambots/teambots.pdf

[11] Amar Banerjee and Venkatesh Choppella. 2023. Knowledge driven synthesis using resource-capability semantics for control software design. *IEEE Access* 11 (2023), 52527–52539.

[12] Ankica Barišić, Ivan Ruchkin, Dušan Savić, Mustafa Abshir Mohamed, Rima Al-Ali, Letitia W. Li, Hana Mkaouar, Raheleh Eslampanah, Moharram Challenger, Dominique Blouin, et al. 2022. Multi-paradigm modeling for cyber–physical systems: A systematic mapping review. *Journal of Systems and Software* 183 (2022), 111081.

[13] Amel Bennaceur, Carlo Ghezzi, Kenji Tei, Timo Kehrer, Danny Weyns, Radu Calinescu, Schahram Dustdar, Zhenjiang Hu, Shinichi Honiden, Fuyuki Ishikawa, et al. 2019. Modelling and analysing resilient cyber-physical systems. In *Proceedings of the 14th Symposium on Software Engineering for Adaptive and Self-Managing Systems*. Retrieved from https://hal.archives-ouvertes.fr/hal-02104308

[14] D. Biermann, J. Gausemeier, H.-P. Heim, S. Hess, M. Petersen, A. Ries, and T. Wagner. 2014. A framework for the computer-aided planning and optimisation of manufacturing processes for components with functional graded properties. In *Proceedings of the AIP Conference Proceedings*, Vol. 1593, AIP, 762–765.

[15] Marcello Bonfe, Cesare Fantuzzi, and Cristian Secchi. 2013. Design patterns for model-based automation software design and implementation. *Control Engineering Practice* 21, 11 (2013), 1608–1619.

[16] Rodney Brooks. 1986. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation* 2, 1 (1986), 14–23.

[17] Davide Brugali and Azamat Shakhimardanov. 2010. Component-based robotic engineering (part II). *IEEE Robotics & Automation Magazine* 17, 1 (2010), 100–112.

[18] Yuriy Brun, Ron Desmarais, Kurt Geihs, Marin Litoiu, Antonia Lopes, Mary Shaw, and Michael Smit. 2013. A design space for self-adaptive systems. In *Software Engineering for Self-Adaptive Systems II*. Springer, 33–50.

[19] Herman Bruyninckx, Markus Klotzbücher, Nico Hochgeschwender, Gerhard Kraetzschmar, Luca Gherardi, and Davide Brugali. 2013. The BRICS component model: A model-based development paradigm for complex robotics software systems. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. ACM, 1758–1764.

[20] Tomas Bures, Danny Weyns, Christian Berger, Stefan Biffl, Marian Daun, Thomas Gabor, David Garlan, Ilias Gerostathopoulos, Christine Julien, Filip Krikava, et al. 2015. Software engineering for smart cyber-physical systems–towards a research agenda: Report on the First International Workshop on Software Engineering for Smart CPS. *ACM SIGSOFT Software Engineering Notes* 40, 6 (2015), 28–32.

[21] Isidro Calvo, Eva Portillo, Oier García de Albéniz, Aintzane Armentia, Marga Marcos, Elisabet Estévez, Ricardo Marau, Luis Almeida, and Paulo Pedreiras. 2012. Towards an infrastructure model for composing and reconfiguring cyber-physical systems. In *Proceedings of the International Conference on Ubiquitous Computing and Ambient Intelligence*. Springer, 282–289.

[22] Arquimedes Canedo, Mohammad Abdullah Al Faruque, and Jan H. Richter. 2014. Multi-disciplinary integrated design automation tool for automotive cyber-physical systems. In *Proceedings of the 2014 Design, Automation & Test in Europe Conference & Exhibition (DATE '14)*. IEEE, 1–2.

[23] Elizabeth Cha, Anca D. Dragan, and Siddhartha S. Srinivasa. 2015. Perceived robot capability. In *Proceedings of the 2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN '15)*. IEEE, 541–548.

[24] Craig B. Chapman and Martyn Pinfold. 1999. Design engineering—A need to rethink the solution using knowledge based engineering. *Knowledge-Based Systems* 12, 5–6 (1999), 257–267.

[25] Craig B. Chapman and Martyn Pinfold. 2001. The application of a knowledge based engineering approach to the rapid design and analysis of an automotive structure. *Advances in Engineering Software* 32, 12 (2001), 903–912.

[26] Subhrojyoti Roy Chaudhuri, Amar Banerjee, N. Swaminathan, Venkatesh Choppella, Arpan Pal, and Balamurali P. (Eds.). 2019. A knowledge centric approach to conceptualizing robotic solutions. In *Proceedings of the 12th Innovations in Software Engineering Conference (formerly known as India Software Engineering Conference) (ISEC '19)*. ACM. DOI: https://doi.org/10.1145/3299771.3299782

[27] Sandeep Chopra, Harish C. Sharma, Pradeep Semwal, and Sanjay Sharma. 2014. Software model for quality controlled component based software system. *International Journal of Advanced Research in Computer Science and Software Engineering* 4, 8 (2014).

[28] Kwok Tai Chui, Brij B. Gupta, Jiaqi Liu, Varsha Arya, Nadia Nedjah, Ammar Almomani, and Priyanka Chaurasia. 2023. A survey of internet of things and cyber-physical systems: standards, algorithms, applications, security, challenges, and future directions. *Information* 14, 7 (2023), 388.

[29] Federico Ciccozzi, Ivica Crnkovic, Davide Di Ruscio, Ivano Malavolta, Patrizio Pelliccione, and Romina Spalazzese. 2017. Model-driven engineering for mission-critical IoT systems. *IEEE Software* 34, 1 (2017), 46–53.

[30] Jake R. Conway, Alexander Lex, and Nils Gehlenborg. 2017. UpSetR: An R package for the visualization of intersecting sets and their properties. *Bioinformatics* 33, 18 (2017), 2938–2940.

[31] Lucas Cordeiro, Carlos Mar, Eduardo Valentin, Fabiano Cruz, Daniel Patrick, Raimundo Barreto, and Vicente Lucena. 2008. A platform-based software design methodology for embedded control systems: An agile toolkit. In *Proceedings of the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS '08)*. IEEE, 408–417.

[32] Maíra Martins Da Silva, Olivier Brüls, Wim Desmet, and Hendrik Van Brussel. 2009. Integrated structure and control design for mechatronic systems with configuration-dependent dynamics. *Mechatronics* 19, 6 (2009), 1016–1025.

[33] Rogério De Lemos, Holger Giese, Hausi A. Müller, Mary Shaw, Jesper Andersson, Marin Litoiu, Bradley Schmerl, Gabriel Tamura, Norha M. Villegas, Thomas Vogel, et al. 2013. Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems II*. Springer, 1–32.

[34] Saadia Dhouib, Selma Kchir, Serge Stinckwich, Tewfik Ziadi, and Mikal Ziane. 2012. RobotML, a domain-specific language to design, simulate and deploy robotic applications. In *Proceedings of the International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 149–160.

[35] Giovanni Di Orio, José Barata, Carlos Sousa, and Luis Flores. 2013. Control system software design methodology for automotive industry. In *Proceedings of the 2013 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 3848–3853.

[36] Alonso Diego, Vicente-Chicote Cristina, Ortiz Francisco, Pastor Juan, and Álvarez Bárbara. 2010. V3cmm: A 3-view component meta-model for model-driven robotic software development. *Journal of Software Engineering in Robotics* 1, 1 (2010), 3–17.

[37] Pengfei Duan, Ying Zhou, Xufang Gong, and Bixin Li. 2018. A systematic mapping study on the verification of cyber-physical systems. *IEEE Access* 6 (2018), 59043–59064.

[38] Clément Duffau, Bartosz Grabiec, and Mireille Blay-Fornarino. 2017. Towards embedded system agile development challenging verification, validation and accreditation: Application in a healthcare company. In *Proceedings of the 2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW '17)*. IEEE, 82–85.

[39] Ulrik Eklund and Jan Bosch. 2012. Applying agile development in mass-produced embedded systems. In *Proceedings of the International Conference on Agile Software Development*. Springer, 31–46.

[40] Ahmed Elkhodary, Naeem Esfahani, and Sam Malek. 2010. FUSION: A framework for engineering self-tuning self-adaptive software systems. In *Proceedings of the 18th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 7–16.

[41] Elisabet Estévez, Alejandro Sánchez-García, Javier Gámez-García, Juan Gómez-Ortega, and Silvia Satorres-Martínez. 2016. A novel model-driven approach to support development cycle of robotic systems. *The International Journal of Advanced Manufacturing Technology* 82, 1–4 (2016), 737–751.

[42] Alessandro Farinelli, Giorgio Grisetti, and Luca Iocchi. 2006. Design and implementation of modular software for programming mobile robots. *International Journal of Advanced Robotic Systems* 3, 1 (2006), 7.

[43] Antonio Filieri, Martina Maggio, Konstantinos Angelopoulos, Nicolás D'Ippolito, Ilias Gerostathopoulos, Andreas Berndt Hempel, Henry Hoffmann, Pooyan Jamshidi, Evangelia Kalyvianaki, Cristian Klein, et al. 2015. Software engineering meets control theory. In *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE Press, 71–82.

[44] Antonio Filieri and Giordano Tamburrelli. 2013. Probabilistic verification at runtime for self-adaptive systems. In *Assurances for Self-Adaptive Systems*. Springer, 30–59.

[45] John Fitzgerald, Carl Gamble, Peter Gorm Larsen, Kenneth Pierce, and Jim Woodcock. 2015. Cyber-physical systems design: Formal foundations, methods and integrated tool chains. In *Proceedings of the 2015 IEEE/ACM 3rd FME Workshop on Formal Methods in Software Engineering*. IEEE, 40–46.

[46] Elvis Foster and Bradford Towle Jr. 2021. *Software Engineering: A Methodical Approach*. Auerbach Publications.

[47] Gianpiero Francesca, Manuele Brambilla, Arne Brutschy, Vito Trianni, and Mauro Birattari. 2014. AutoMoDe: A novel approach to the automatic design of control software for robot swarms. *Swarm Intelligence* 8, 2 (2014), 89–112.

[48] Alejandro Germán Frank, Lucas Santos Dalenogare, and Néstor Fabián Ayala. 2019. Industry 4.0 technologies: Implementation patterns in manufacturing companies. *International Journal of Production Economics* 210 (2019), 15–26.

[49] P. Fritzson. 2011. A cyber-physical modeling language and the OpenModelica environment. In *Proceedings of the International Wireless Communications and Mobile Computing Conference*, 4–8.

[50] David Gay, Philip Levis, Robert Von Behren, Matt Welsh, Eric Brewer, and David Culler. 2014. The nesC language: A holistic approach to networked embedded systems. *ACM SIGPLAN Notices* 49, 4 (2014), 41–51.

[51] Giovanni Godena, Tomaž Lukman, and Gregor Kandare. 2013. A new approach to control systems software development. In *Case Studies in Control*. Springer, 363–406.

[52] Iris Gräßler, Dominik Wiechel, Daniel Roesmann, and Henrik Thiele. 2021. V-model based development of cyber-physical systems and cyber-physical production systems. *Procedia CIRP* 100 (2021), 253–258.

[53] Zuguang Gu. 2022. Complex heatmap visualization. *Imeta* 1, 3 (2022), e43.

[54] Zuguang Gu, Roland Eils, and Matthias Schlesner. 2016. Complex heatmaps reveal patterns and correlations in multidimensional genomic data. *Bioinformatics* 32, 18 (2016), 2847–2849.

[55] Zuguang Gu, Roland Eils, Matthias Schlesner, and Naveed Ishaque. 2018. EnrichedHeatmap: An R/Bioconductor package for comprehensive visualization of genomic signal associations. *BMC Genomics* 19 (2018), 1–7.

[56] Arne Haber. 2016. *MontiArc-Architectural Modeling and Simulation of Interactive Distributed Systems*. Vol. 24, Shaker Verlag GmbH.

[57] P. J. Hall. 2005. The square kilometre array: An international engineering perspective. In *The Square Kilometre Array: An Engineering Perspective*. Springer, 5–16.

[58] Svein Hallsteinsen, Kurt Geihs, Nearchos Paspallis, Frank Eliassen, Geir Horn, Jorge Lorenzo, Alessandro Mamelli, and George Angelos Papadopoulos. 2012. A development framework and methodology for self-adapting applications in ubiquitous computing environments. *Journal of Systems and Software* 85, 12 (2012), 2840–2859.

[59] Reinhard Hametner, Dietmar Winkler, Thomas Östreicher, Stefan Biffl, and Alois Zoitl. 2010. The adaptation of test-driven software processes to industrial automation engineering. In *Proceedings of the 2010 8th IEEE International Conference on Industrial Informatics*. IEEE, 921–927.

[60] Robert Harrison, Daniel Vera, and Bilal Ahmad. 2016. Engineering methods and tools for cyber–physical automation systems. *Proceedings of the IEEE* 104, 5 (2016), 973–985.

[61] Thomas Hartmann, François Fouquet, Jacques Klein, Grégory Nain, and Yves Le Traon. 2014. Reactive security for smart grids using models@ run. Time-based simulation and reasoning. In *Proceedings of the International Workshop on Smart Grid Security*. Springer, 139–153.

[62] Taoufik Ben Hassine, Oualid Khayati, and Henda Ben Ghezala. 2017. An IoT domain meta-model and an approach to software development of IoT solutions. In *Proceedings of the 2017 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC '17)*. IEEE, 32–37.

[63] David Hästbacka and Seppo Kuikka. 2013. Semantics enhanced engineering and model reasoning for control application development. *Multimedia Tools and Applications* 65, 1 (2013), 47–62.

[64] David Hästbacka, Timo Vepsäläinen, and Seppo Kuikka. 2011. Model-driven development of industrial process control applications. *Journal of Systems and Software* 84, 7 (2011), 1100–1113.

[65] Shyamanta M. Hazarika and Uday Shanker Dixit. 2018. Robotics: History, trends, and future directions. In *Introduction to Mechanical Engineering*. Springer, Cham, 213–239.

[66] Jie He, Yishuang Geng, Yadong Wan, Shen Li, and Kaveh Pahlavan. 2013. A cyber physical test-bed for virtualization of RF access environment for body sensor network. *IEEE Sensors Journal* 13, 10 (2013), 3826–3836.

[67] Jon Hemmerdinger and David Kaminski-Morrow. 2019. ET302 crew 'could not control aircraft': Ethiopian transport ministry says pilots were unable to prevent uncommanded nose-down conditions in 737 Max. *Flight International*.

[68] Steffen Henning, Oliver Niggemann, Jens Otto, and Sebastian Schriegel. 2014. A descriptive engineering approach for cyber-physical systems. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA '14)*. IEEE, 1–4.

[69] Nico Hochgeschwender, Luca Gherardi, Azamat Shakhirmardanov, Gerhard K. Kraetzschmar, Davide Brugali, and Herman Bruyninckx. 2013. A model-based approach to software deployment in robotics. In *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '13)*. IEEE, 3907–3914.

[70] Florian Hofer. 2018. Architecture, technologies and challenges for cyber-physical systems in industry 4.0: A systematic mapping study. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 1–10.

[71] Nicholas J. Horton and Ken Kleinman. 2015. *Using R and RStudio for Data Management, Statistical Analysis, and Graphics*. CRC Press.

[72] Huang-Ming Huang, Terry Tidwell, Christopher Gill, Chenyang Lu, Xiuyu Gao, and Shirley Dyke. 2010. Cyber-physical systems for real-time hybrid structural testing: A case study. In *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*. ACM, 69–78.

[73] Georg Jäger, Christian A. Mueller, Madhura Thosar, Sebastian Zug, and Andreas Birk. 2018. Towards robot-centric conceptual knowledge acquisition. arXiv:1810.03583. Retrieved from https://arxiv.org/abs/1810.03583

[74] Choulsoo Jang, Seung-Ik Lee, Seung-Woog Jung, Byoungyoul Song, Rockwon Kim, Sunghoon Kim, and Cheol-Hoon Lee. 2010. OPRoS: A new component-based robot software platform. *ETRI Journal* 32, 5 (2010), 646–656.

[75] Dayang N. A. Jawawi, Rosbi Mamat, and Safaai Deris. 2007. A component-oriented programming for embedded mobile robot software. *International Journal of Advanced Robotic Systems* 4, 3 (2007), 40.

[76] Miguel Jiménez, Norha M. Villegas, Gabriel Tamura, and Hausi A. Müller. 2017. Deployment specification challenges in the context of large scale systems. In *Proceedings of the 27th Annual International Conference on Computer Science and Software Engineering (CASCON '17)*. IBM Corp., Riverton, NJ, 220–226. Retrieved from http://dl.acm.org/citation.cfm?id=3172795.3172821

[77] Aaron Kane, Thomas Fuhrman, and Philip Koopman. 2014. Monitor based oracles for cyber-physical system testing: Practical experience report. In *Proceedings of the 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 148–155.

[78] Gabor Karsai, Janos Sztipanovits, Akos Ledeczi, and Ted Bapty. 2003. Model-integrated development of embedded software. *Proceedings of the IEEE* 91, 1 (2003), 145–164.

[79] Muhammed Cagri Kaya, Alperen Eroglu, Alper Karamanlioglu, Ertan Onur, Bedir Tekinerdogan, and Ali H. Dogru. 2019. Runtime adaptability of ambient intelligence systems based on component-oriented approach. In *Guide to Ambient Intelligence in the IoT Environment*. Springer, 69–92.

[80] Tariq M. King, Alain E. Ramirez, Rodolfo Cruz, and Peter J. Clarke. 2007. An integrated self-testing framework for autonomic computing systems. *Journal of Computers* 2, 9 (2007), 37–49.

[81] Nathan Koenig and Andrew Howard. 2004. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '04)(IEEE Cat. No. 04CH37566)*, Vol. 3, IEEE, 2149–2154.

[82] Hitoshi Komoto and Tetsuo Tomiyama. 2012. A framework for computer-aided conceptual design and its application to system architecting of mechatronics products. *Computer-Aided Design* 44, 10 (2012), 931–946.
[83] Xabier Larrucea, Annie Combelles, John Favaro, and Kunal Taneja. 2017. Software engineering for the Internet of Things. *IEEE Software* 34, 1 (2017), 24–28.
[84] Jay Lee, Behrad Bagheri, and Hung-An Kao. 2015. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters* 3 (2015), 18–23.
[85] Jae-Hee Lim, Suk-Hoon Song, Jung-Rye Son, Tae-Yong Kuc, Hong-Seong Park, and Hong-Seok Kim. 2010. An automated test method for robot platform and its components. *International Journal of Software Engineering and Its Applications* 4, 3 (2010), 9–18.
[86] Oscar Ljungkrantz, Knut Akesson, Martin Fabian, and Chengyin Yuan. 2010. Formal specification and verification of industrial control logic components. *IEEE Transactions on Automation Science and Engineering* 7, 3 (2010), 538–548.
[87] Damian M. Lyons, Ronald C. Arkin, Paramesh Nirmal, Shu Jiang, and T.-M. Liu. 2013. A software tool for the design of critical robot missions with performance guarantees. *Procedia Computer Science* 16 (2013), 888–897.
[88] Douglas C. MacKenzie and Ronald C. Arkin. 1994. Formal specification for behavior-based mobile robots. In *Mobile Robots VIII*, Vol. 2058, International Society for Optics and Photonics, 94–105.
[89] M. Masin, Francesca Palumbo, H. Myrhaug, J. A. de Oliveira Filho, M. Pastena, Maxime Pelcat, Luigi Raffo, Francesco Regazzoni, A. A. Sanchez, A. Toffetti, et al. 2017. Cross-layer design of reconfigurable cyber-physical systems. In *Proceedings of the Conference on Design, Automation & Test in Europe*. European Design and Automation Association, 740–745.
[90] Giorgio Metta, Paul Fitzpatrick, and Lorenzo Natale. 2006. YARP: Yet another robot platform. *International Journal of Advanced Robotic Systems* 3, 1 (2006), 8.
[91] Samoel Mirachi, Valdir da Costa Guerra, Adilson Marques da Cunha, Luiz Alberto Vieira Dias, and Emilia Villani. 2017. Applying agile methods to aircraft embedded software: An experimental analysis. *Software: Practice and Experience* 47, 11 (2017), 1465–1484.
[92] Mustafa Abshir Mohamed, Moharram Challenger, and Geylani Kardas. 2020. Applications of model-driven engineering in cyber-physical systems: A systematic mapping study. *Journal of Computer Languages* 59 (2020), 100972.
[93] Brice Morin, Nicolas Harrand, and Franck Fleurey. 2017. Model-based software engineering to tame the IoT jungle. *IEEE Software* 34, 1 (2017), 30–36.
[94] Hausi Muller, John Mylopoulos, and Marin Litoiu. 2015. Engineering cyber physical systems. In *Proceedings of the 25th Annual International Conference on Computer Science and Software Engineering (CASCON '15)*. IBM Corp., Riverton, NJ, 328–332. Retrieved from http://dl.acm.org/citation.cfm?id=2886444.2886514
[95] H. A. Müller. 2013. Software engineering for the industrial Internet: Situation-aware smart applications. In *Proceedings of the 2013 15th IEEE International Symposium on Web Systems Evolu*tion (WSE '13), 1–1. DOI: https://doi.org/10.1109/WSE.2013.6642408
[96] Viet Hoa Nguyen, François Fouquet, Noël Plouzeau, and Olivier Barais. 2012. A process for continuous validation of self-adapting component based systems. In *Proceedings of the 7th Workshop on Models@ run. time.* ACM, 32–37.
[97] Tim Niemueller, Alexander Ferrein, Daniel Beck, and Gerhard Lakemeyer. 2010. Design principles of the component-based robot software framework fawkes. In *Proceedings of the International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 300–311.
[98] Oliver Niggemann and Björn Kroll. 2014. On the applicability of model based software development to cyber physical production systems. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA '14)*. IEEE, 1–4.
[99] Pierluigi Nuzzo. 2015. *Compositional Design of Cyber-Physical Systems Using Contracts*. Ph.D. Dissertation. EECS Department, University of California, Berkeley. Retrieved from http://www2.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-189.html
[100] Jens Otto, Steffen Henning, and Oliver Niggemann. 2014. Why cyber-physical production systems need a descriptive engineering approach–A case study in plug & produce. *Procedia Technology* 15 (2014), 295–302.
[101] Seno Darmawan Panjaitan. 2008. *Development Process for Distributed Automation Systems Based on Elementary Mechatronic Functions*. Shaker.
[102] Puneet Patwari, Amar Banerjee, Subhrojyoti Roy Chaudhuri, and Swaminathan Natarajan. 2016. Learning's from developing a domain specific engineering environment for control systems. In *Proceedings of the 9th India Software Engineering Conference*. ACM, 177–183.
[103] Kai Petersen, Sairam Vakkalanka, and Ludwik Kuzniarz. 2015. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology* 64 (2015), 1–18.
[104] João Pimentel, Márcia Lucena, Jaelson Castro, Carla Silva, Emanuel Santos, and Fernanda Alencar. 2012. Deriving software architectural models from requirements models for adaptive systems: The STREAM-A approach. *Requirements Engineering* 17, 4 (2012), 259–281.

[105] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. 2009. ROS: An open-source robot operating system. In *Proceedings of the ICRA Workshop on Open Source Software*, Vol. 3, 5.

[106] Muthu Ramachandran. 2018. Secure software development of cyber-physical and IoT systems. In *Encyclopedia of Information Science and Technology (4th ed.)*. IGI Global, 7525–7538.

[107] Arunkumar Ramaswamy, Bruno Monsuez, and Adriana Tapus. 2014. Model-driven software development approaches in robotics research. In *Proceedings of the 6th International Workshop on Modeling in Software Engineering*. ACM, 43–48.

[108] Laurent Réveillère. 2011. *Building Efficient Distributed Systems: A Domain-Specific Language Based Approach*. Ph.D. Dissertation. Université Sciences et Technologies-Bordeaux I.

[109] Luis Riazuelo, Moritz Tenorth, Daniel Di Marco, Marta Salas, Dorian Gálvez-López, Lorenz Mösenlechner, Lars Kunze, Michael Beetz, Juan D. Tardós, Luis Montano, et al. 2015. RoboEarth semantic mapping: A cloud enabled knowledge-based approach. *IEEE Transactions on Automation Science and Engineering* 12, 2 (2015), 432–443.

[110] Jan Oliver Ringert, Alexander Roth, Bernhard Rumpe, and Andreas Wortmann. 2015. Code generator composition for model-driven engineering of robotics component & connector systems. arXiv:1505.00904. Retrieved from https://arxiv.org/abs/1505.00904

[111] Jan Oliver Ringert, Bernhard Rumpe, and Andreas Wortmann. 2014. A case study on model-based development of robotic systems using MontiArc with embedded automata. arXiv:1408.5692. Retrieved from https://arxiv.org/abs/1408.5692

[112] Kay Römer. 2010. Threads2Events: An automatic code generation approach. In *Proceedings of the 6th Workshop on Hot Topics in Embedded Networked Sensors*, 1–5.

[113] Nilay K. Roy, Michael A. Ridge, Scott E. Lennox, Rami Mangoubi, and Murali V. Chaparala. 2019. Distributed system for management and control of aerial vehicle air traffic. US Patent App. 16/153241.

[114] Stefan Runde, Alexander Fay, and Wagner-Otto Wutzke. 2009. Knowledge-based requirement-engineering of building automation systems by means of Semantic Web technologies. In *Proceedings of the 2009 7th IEEE International Conference on Industrial Informatics*. IEEE, 267–272.

[115] Christian Schlegel, Andreas Steck, Davide Brugali, and Alois Knoll. 2010. Design abstraction and processes in robotics: From code-driven to model-driven engineering. In *Proceedings of the International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 324–335.

[116] Christian Schlegel, Andreas Steck, and Alex Lotz. 2012. Robotic software systems: From code-driven to model-driven software development. In *Robotic Systems-Applications, Control and Programming*. InTech.

[117] Syed Imran Shafiq, Cesar Sanin, Edward Szczerbicki, and Carlos Toro. 2015. Virtual engineering object/virtual engineering process: A specialized form of cyber physical system for Industrie 4.0. *Procedia Computer Science* 60 (2015), 1146–1155.

[118] Azamat Shakhimardanov, Nico Hochgeschwender, and Gerhard K. Kraetzschmar. 2010. Component models in robotics software. In *Proceedings of the 10th Performance Metrics for Intelligent Systems Workshop*. ACM, 82–87.

[119] Seyed Mahdi Shavarani, Mazyar Ghadiri Nejad, Farhood Rismanchian, and Gokhan Izbirak. 2018. Application of hierarchical facility location problem for optimization of a drone delivery system: A case study of Amazon prime air in the city of San Francisco. *The International Journal of Advanced Manufacturing Technology* 95, 9–12 (2018), 3141–3153.

[120] Stepan Shevtsov, Danny Weyns, and Martina Maggio. 2019. Self-adaptation of software using automatically generated control-theoretical solutions. In *Engineering Adaptive Software Systems*. Springer, 35–55.

[121] Lenardo C. Silva, Mirko Perkusich, Frederico M. Bublitz, Hyggo O. Almeida, and Angelo Perkusich. 2014. A model-based architecture for testing medical cyber-physical systems. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*. ACM, 25–30.

[122] Reid Simmons and David Apfelbaum. 1998. A task description language for robot control. In *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 3, IEEE, 1931–1937.

[123] Majid Sorouri. 2014. *A Compositional Approach to Control Software Design of Automation Systems Based on Mechatronic Modularity*. Ph.D. Dissertation. ResearchSpace@ Auckland.

[124] Paul Soulier, Depeng Li, and John R. Williams. 2015. A survey of language-based approaches to cyber-physical and embedded system development. *Tsinghua Science and Technology* 20, 2 (2015), 130–141.

[125] Samir K. Srivastava. 2016. Industry 4.0. BHU Engineer's Alumni, Lucknow.

[126] Timothy J. Starkweather, Ronald J. Lebel, Varaz Shahmirian, Philip T. Weiss, and David J. Marsh. 2004. Ambulatory medical apparatus and method having telemetry modifiable control software. US Patent 6694191.

[127] Thomas Strasser, Martijn Rooker, Gerhard Ebenhofer, Alois Zoitl, Christoph Sunder, Antonio Valentini, and Allan Martel. 2008. Structuring of large scale distributed control programs with IEC 61499 subapplications and a hierarchical plant structure model. In *Proceedings of the 2008 IEEE International Conference on Emerging Technologies and Factory Automation*. IEEE, 934–941.

[128] Thomas Strasser, Christoph Sunder, and Antonio Valentini. 2008. Model-driven embedded systems design environment for the industrial automation sector. In *Proceedings of the 2008 6th IEEE International Conference on Industrial Informatics*. IEEE, 1120–1125.
[129] Moritz Tenorth and Michael Beetz. 2013. KnowRob: A knowledge processing infrastructure for cognition-enabled robots. *The International Journal of Robotics Research* 32, 5 (2013), 566–590.
[130] Moritz Tenorth, Alexander Clifford Perzylo, Reinhard Lafrenz, and Michael Beetz. 2012. The roboearth language: Representing and exchanging knowledge about actions, objects, and environments. In *Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA '12)*. IEEE, 1284–1289.
[131] Kleanthis Thramboulidis. 2015. A cyber–physical system-based approach for industrial automation systems. *Computers in Industry* 72 (2015), 92–102.
[132] Kleanthis C. Thramboulidis. 2004. Using UML in control and automation: A model driven approach. In *Proceedings of the 2nd IEEE International Conference on Industrial Informatics (INDIN '04)*. IEEE, 587–593.
[133] Terry Tidwell, Xiuyu Gao, Huang-Ming Huang, Chenyang Lu, Shirley Dyke, and Christopher Gill. 2009. Towards configurable real-time hybrid structural testing: A cyber-physical system approach. In *Proceedings of the 2009 IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*. IEEE, 37–44.
[134] Nikolaus Vahrenkamp, Mirko Wächter, Manfred Kröhnert, Kai Welke, and Tamim Asfour. 2015. The robot software framework ArmarX. *Information Technology* 57, 2 (2015), 99–111.
[135] Birgit Vogel-Heuser, Daniel Schütz, Timo Frank, and Christoph Legat. 2014. Model-driven engineering of manufacturing automation software projects–A SysML-based approach. *Mechatronics* 24, 7 (2014), 883–897.
[136] Valeriy Vyatkin and Hans-Michael Hanisch. 2003. Verification of distributed control systems in intelligent manufacturing. *Journal of Intelligent Manufacturing* 14, 1 (2003), 123–136.
[137] Jiafu Wan, Hui Suo, Hehua Yan, and Jianqi Liu. 2011. A general test platform for cyber-physical systems: Unmanned vehicle with wireless sensor network navigation. *Procedia Engineering* 24 (2011), 123–127.
[138] Lihui Wang, Martin Törngren, and Mauro Onori. 2015. Current status and advancement of cyber-physical systems in manufacturing. *Journal of Manufacturing Systems* 37 (2015), 517–527.
[139] Amanda Whitbrook. 2009. *Programming Mobile Robots with Aria and Player: A Guide to C++ Object-Oriented Control*. Springer Science & Business Media.
[140] Johannes Wienke, Dennis Wigand, Norman Koster, and Sebastian Wrede. 2018. Model-based performance testing for robotics software components. In *Proceedings of the 2018 2nd IEEE International Conference on Robotic Computing (IRC '18)*. IEEE, 25–32.
[141] Martin Wirsing, Matthias Hölzl, Nora Koch, and Philip Mayer. 2015. *Software Engineering for Collective Autonomic Systems: The ASCENS Approach*. Vol. 8998, Springer.
[142] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering*. Vol. 236, Springer.
[143] Andreas Wortmann and Martin Beet. 2016. Domain specific languages for efficient satellite control software development. In *Proceedings of the Data Systems in Aerospace (DASIA '16)*, Vol. 736.
[144] Hongyan Xia, Jonathan Woodruff, Hadrien Barral, Lawrence Esswood, Alexandre Joannou, Robert Kovacsics, David Chisnall, Michael Roe, Brooks Davis, Edward Napierala, et al. 2018. CheriRTOS: A capability model for embedded devices. In *Proceedings of the International Conference on Computer Design*, 92–99.
[145] Jeffrey Yan and Valeriy Vyatkin. 2013. Distributed software architecture enabling peer-to-peer communicating controllers. *IEEE Transactions on Industrial Informatics* 9, 4 (2013), 2200–2209.
[146] Chia-Han Yang, Valeriy Vyatkin, and Cheng Pang. 2014. Model-driven development of control software for distributed automation: A survey and an approach. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 44, 3 (2014), 292–305.
[147] Klemen Zagar, Anze Vodovnik, and J. Stefan. 2002. Program generators and control system software development. In *Proceedings of 4th International Conference of Personal Computers and Particle Accelerator Controls (PCaPAC2002)*. Retrieved from http://www.lnf.infn.it/conference/pcapac2002/TALK/WE-07/WE-07.pdf
[148] Maryam Zahid, Irum Inayat, Maya Daneva, and Zahid Mehmood. 2021. Security risks in cyber physical systems—A systematic mapping study. *Journal of Software: Evolution and Process* 33, 9 (2021), e2346.
[149] Ji Zhang and Betty H. C. Cheng. 2006. Model-based development of dynamically adaptive software. In *Proceedings of the 28th International Conference on Software engineering*. ACM, 371–380.
[150] Lichen Zhang, Jifeng He, and Wensheng Yu. 2013. Test case generation from formal models of cyber physical system. *International Journal of Hybrid Information Technology* 6, 3 (2013), 15–24.
[151] Quanyan Zhu and Tamer Basar. 2015. Game-theoretic methods for robustness, security, and resilience of cyberphysical control systems: Games-in-games principle for optimal cross-layer resilient control systems. *IEEE Control Systems Magazine* 35, 1 (2015), 46–65.

Appendix

Table A1. Engineering Methods Used in Articles

Engineering Methods	Studies
Waterfall	[13, 19, 20, 22, 29, 33, 50, 60, 62, 68, 74, 78, 83, 93, 94, 99–101, 123, 131, 135, 141, 146]
Agile	[38, 39, 91]

Table A2. Engineering Phases Targeted in Articles

Engineering Phases	Studies
Requirements	[3, 23, 104, 114]
Design	[4, 9, 13–15, 17–19, 21, 22, 25, 31, 32, 34–36, 40, 43, 45, 47, 49, 51, 56, 63, 64, 68, 74, 75, 82, 84, 87, 89, 90, 95, 99, 104, 111, 115–118, 123, 127, 128, 132, 134, 143, 144, 149]
Implementation Methods	[2, 9, 10, 16, 19, 27, 34, 41, 42, 47, 51, 58, 62, 64, 75, 78, 88, 90, 93, 95, 97, 106, 108, 110, 112, 115, 118, 120, 122, 124, 131, 134, 139, 146, 147]
Verification Methods	[7, 13, 38, 44, 45, 59, 61, 63, 66, 72, 77, 80, 85–88, 94, 96, 117, 121, 134, 136, 137, 140, 150]
Integration Methods	[32, 76, 145]
Operations Methods	[2, 13, 40, 44, 69, 72, 73, 77, 79, 96, 134]

Table A3. Engineering Paradigms Used in Articles

Engineering Paradigms	Studies
Model Driven	[1–4, 9, 13, 14, 17–19, 21–23, 27, 31, 32, 34, 36, 40, 45, 49, 51, 56, 59, 62, 64, 66, 68, 69, 74, 78, 80, 85–87, 98–100, 110, 111, 115–117, 121, 123, 127, 128, 132, 134, 135, 136, 140, 144, 146, 147, 149]
Code Driven	[2, 7, 10, 16, 42, 44, 47, 50, 58, 72, 75–77, 88, 90, 93–97, 101, 106, 112, 118, 120, 124, 131, 137, 139, 143, 150]
Knowledge Driven	[25, 63, 114, 117]

Table A4. Research Objectives Targeted in Articles

Research Objective	Studies
Programming Framework	[14, 16, 35, 42, 58, 75, 93, 97, 118, 122, 124, 131, 139, 143]
Tool	[4, 9, 10, 19, 22, 25, 31, 36, 40, 51, 66, 74, 75, 80, 87, 90, 112, 115–117, 134, 137, 140, 147]
Concept	[1–3, 17, 18, 20, 23, 33, 39, 43, 45, 59, 61, 64, 72, 73, 76–78, 82, 85, 91, 94, 96, 100, 112, 114, 120, 128, 132, 135, 146, 151]
Language	[34, 49, 50, 108]
Formalism	[7, 44, 47, 79, 86, 88, 99, 123, 136, 149, 150]
Architecture	[4, 13, 19, 21, 27, 31, 32, 56, 62, 69, 84, 95, 104, 111, 117, 121, 127, 134, 144, 145]

Table A5. Target Domains Targeted in Articles

Target Domain	**Studies**
CPS	[1, 3, 4, 7, 13–15, 18, 20–22, 27, 31, 33, 38, 40, 43, 45, 49, 51, 56, 58–61, 63, 64, 66, 72, 76–78, 80, 84, 86, 91, 94–96, 98, 99, 104, 106, 108, 110, 112, 114, 117, 120, 121, 124, 127, 128, 131–133, 136, 137, 143–147, 149, 150]
Robotics and Mechatronics	[2, 4, 10, 10, 16, 17, 19, 23, 32, 34, 36, 41, 42, 47, 69, 73–75, 82, 85, 87, 88, 90, 96, 97, 101, 110, 111, 115, 116, 118, 122, 123, 134, 139, 140]
IoT	[4, 9, 18, 20, 27, 33, 38, 56, 62, 79, 83, 93, 95, 96, 120, 141, 145, 146]
Automotive	[25, 27, 35, 38, 59, 64, 86, 91]

Table A6. Research Concerns Addressed in Articles

Concern	**Studies**
Safety and Security	[14, 16, 25, 27, 29, 38, 59, 63, 66, 80, 81, 83, 86–88, 93, 95–99, 106, 131, 134, 151]
Realtime Control	[2, 9, 13, 19, 25, 34, 38, 41, 43, 49, 51, 56, 60, 64, 78, 90, 94, 98, 101, 108, 112, 116, 127, 131–134, 139, 141, 146]
Dynamic Adaptation	[10, 18, 21, 33, 40, 58, 61, 79, 89, 104, 120, 123, 149]
Modularity and Interfacing	[4, 15, 17, 19, 21, 22, 23, 32, 41, 42, 45, 56, 62, 64, 68, 74, 75, 78, 84, 97, 100, 110, 111, 115, 116, 118, 121, 123, 127, 128, 141, 143, 144]
Performance	[1, 3, 13, 14, 27, 50, 140, 145]
Fault Detection	[7, 31, 34, 38, 39, 44, 59, 66, 69, 72, 76, 77, 80, 81, 85–87, 96, 99, 106, 121, 133, 136, 137, 150]
Context Awareness	[1, 3, 35, 47, 58, 61–63, 83, 84, 91, 95, 114, 117, 122, 128, 132, 135]
Modularity and Interfacing	[16, 35, 79, 87, 89, 93, 108, 117, 151]

Received 17 January 2024; revised 6 July 2024; accepted 10 November 2024